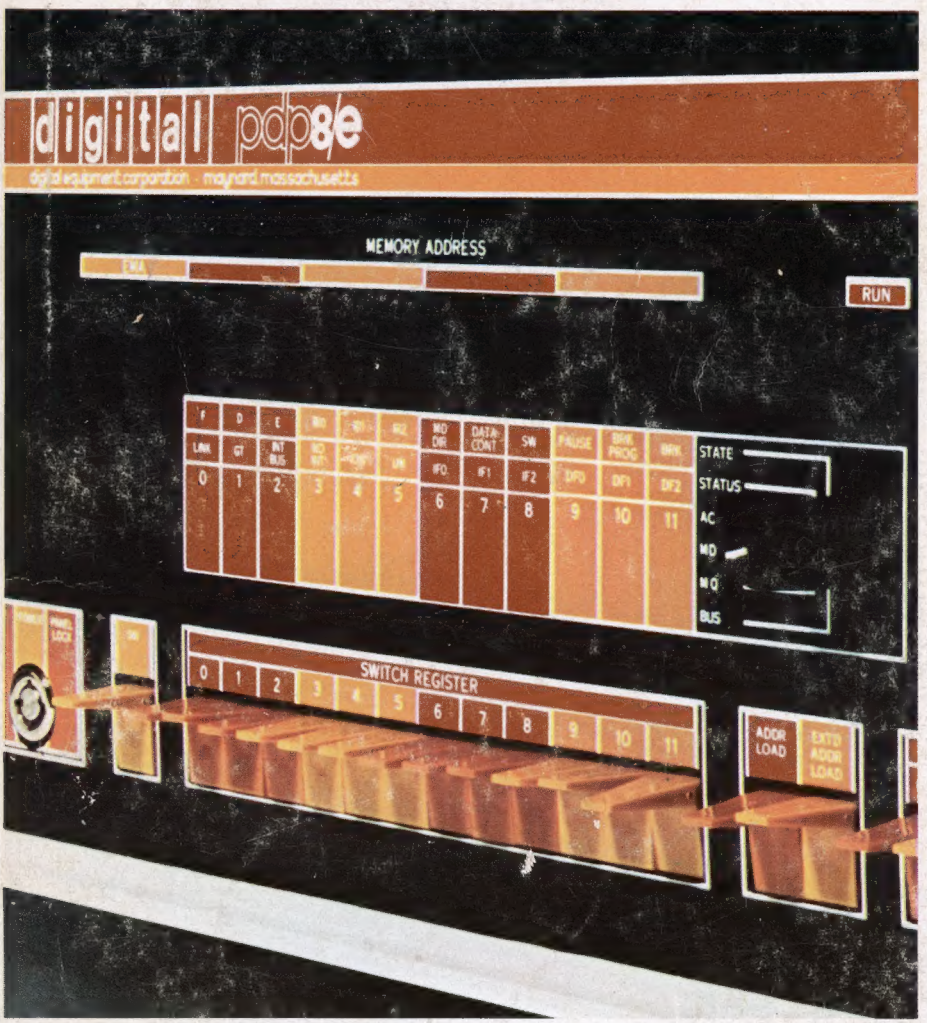
pdp8/e
small computer handbook



digital equipment corporation

digital small computer handbook 1971

part 1

PDP-8/E PROGRAMMED DATA PROCESSOR

part 2

PDP-8/E OPTIONS

part 3

INTERFACING AND INSTALLATION

part 4

DEC SERVICES

digital

pdp8/e

small computer handbook

1971

digital equipment corporation

Copyright © 1970 by
Digital Equipment Corporation

The following are registered trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

DEC
FLIP CHIP
DIGITAL
DIBOL
PDP
FOCAL
COMPUTER LAB

INTRODUCTION

This handbook is another in a series intended to familiarize the user with the Digital Equipment Corporation (DEC) PDP-8 family of small general-purpose computers. It explains the newest member of the family, the PDP-8/E Programmed Data Processor and how it is interfaced with the wide variety of peripheral equipment available.

Major topics are: The Programmed Data Processor, PDP-8/E Options, Interface & Installation, Support Services, and Appendices.

Another member of this series of handbooks is titled: An Introduction to Programming. The programming handbook familiarizes the user with the principles of programming the PDP-8 family of general-purpose computers. Together this handbook and the programming handbook describe the complete hardware and software aspects of the PDP-8 family. A newly released handbook called Programming Languages has been added to the handbook series which contains information on the primary languages for the PDP-8 family such as FOCAL, BASIC, PAL III, MACRO, PAL D, SABR, and 4K FORTRAN.

THE COMPANY

In a little over thirteen years, Digital has become a major force in the electronics industry. The company has grown from three employees and 8,500 square feet of production floor space in a converted woolen mill in Maynard, Massachusetts, to an international corporation employing more than 6,000 people with well over two million square feet of floor space in more than 60 manufacturing, sales, and service facilities around the world. In addition to the corporate headquarters in Maynard, Massachusetts, other manufacturing facilities are located in Westfield and Westminster, Massachusetts. Internationally and outside the continental United States, Digital has manufacturing plants in England, Canada and Puerto Rico.

From its beginnings as a manufacturer of digital modules, the company has now grown to the point where it is the world's largest manufacturing supplier of logic modules and the third largest computer-manufacturer, by number of installations, in the industry. Digital's rise as a leader in the electronics industry began in 1957 with the introduction of the company's line of electronic circuit modules. These solid-state modules were used to build and test other manufacturers' computers. Two years later, Digital introduced its first computer, the PDP-1. The PDP-1 heralded a new concept for the industry—the small, on-line computer. And the PDP-1 was inexpensive—it sold for $120,000 while competitive machines with similar capabilities were selling at over $1 million. But the PDP-1 was more than a data processor; more than just a tool to manipulate data. It was a system that could be connected to all types of instrumentation and equipment for on-line, real-time monitoring control, and analysis. It was a system with which people and machines could interact.

Also, in 1958, Digital introduced the Systems Modules, high-quality, low-cost, solid-state, digital logic circuits on a single printed circuit card.

Today, electronic modules like the ones Digital introduced are used in most electronic equipment, from computers to television sets.

In 1965, Digital announced the first of the FLIP CHIP® module lines. These highly reliable modules include cards for internal computer logic, interfacing, control and analog-to-digital conversion.

In 1963, Digital Equipment Corporation introduced the PDP-5 computer, predecessor of the PDP-8 series. This was followed by the first PDP-8 in 1964, and since that time DEC has added the PDP-8/S, the PDP-8/I, and PDP-8/L. Over this seven year period, considerable improvement has been made, many options have been developed, over 60 peripherals and a variety of programs developed. As each new application need arises, Digital Equipment engineering responds with new equipment; each time further increasing the capability of the PDP-8 Family and making available a wider range of equipment.

Throughout the life span of the PDP-8 Family, DEC has developed more than 1,000 programs for a wide variety of applications. New programs are constantly in development by Digital's Programming Department and the PDP-8 Users. This means that each PDP-8/E user will have a wide variety of programs immediately available to him.

To further enhance the users' capability, the DECUS library contains a wide variety of programs developed by the PDP-8 users. This library is operated by DEC exclusively for customer use. Programs are available for as little as $1.00 each.

USER APPLICATIONS

As a result of the interest in the PDP-8 family more than 8,500 PDP-8 computers are installed all over the world. Applications have increased to embrace almost every discipline known to man and new applications are finding their way to the PDP-8 family every day. Representative categories of these computer applications are given below:

DATA COMMUNICATIONS

- Multiple Data Terminal Systems
- Inter-city Remote Data Systems
- Data Concentrators
- Communications Systems for News Services
- Multi-user Time-Shared Computing
- Message Switching Stations

BIO-MEDICAL

- Environmental Laboratory Response System
- Nuclear Medicine
- General Research
- Clinical Laboratory Systems
- Digital Image and Processing Systems
- Intensive Care Systems
- Multiphasic Health Screening Systems
- Radiation Treatment Planning System
- Cancer Research
- Bone Densitometry

EDUCATION

Computer Assisted General Instruction
Computer Assisted Instructions of Programming Fundamentals
Computer Controlled Audio-Visual Teaching Facility
University Computing Center

INDUSTRIAL

Automatic Drafting System
Automatic Digitizing Systems
High Speed Plotting, Editing and Verification Systems
Automatic Stackers for Warehouses and Stockrooms
Computer Controlled IC, LSI, and MSI Mask Making
Concrete Batching System
Paper Mill Control
Transfer and Assembly Line Control and Monitoring
Data Acquisition and Process Control Systems
Pipe-Line Control
Engineering Simulations

TYPESETTING SYSTEMS

Newspapers
Commercial Typesetters

NUMERICAL CONTROL

Direct Computer Control for Machine Tools
NC Tape Preparation for Various Machine Tools
Digitizing Systems for NC Machine Control

ENTERTAINMENT

Computer Animated Motion Pictures
Television Station Control
Scoreboard and Toteboard Displays
Museum Exhibit Control
Planetarium Control

ENGINEERING

Engineering Calculation
Ship Model Design
Computer Aided Design
Hybrid Simulation

AEROSPACE

Observatory Control and Data Logging
Radio Astrometry
Space Probe Instrumentation, Calibration and Data Reduction
Lunar Sample Analysis
Inertial Guidance Checkout

NAVIGATION

Shipboard Navigation Systems Via Satellite

GEOPHYSICS/MARINE

Oceanographic Data Acquisition
Shipboard Hydrographic Data Acquisition System
Seismic Monitoring
Weather Control Research

TRANSPORTATION

Automated Transit Systems
Traffic Control Systems
Toll Collection and Parking Lot Collection Systems

PHYSICS AND CHEMISTRY

Gas Chromatography
Vehicular Emissions Analyzer
Mass Spectrometry Systems
X-Ray Diffractometers
Pulse Height Analysis
Nuclear Magnetics Resonance
Spectrophotometry

INDUSTRIAL QUALITY CONTROL AND TESTING

Automatic Gauging and Measuring Equipment Systems
Electronic Component Automatic Test and Classification System
Automatic Cable Capacitance Testing System
Automatic Gear Checking System
Noise Measurement System

BUSINESS AND COMMERCE

Accounting and Record Keeping Systems
Business Systems for Newspapers
Mailing Systems
Inventory Control Systems
Automatic Typing Systems
Management Information Systems
Information Systems for Brokers
Reservation Systems
Warehouse Automated Storage and Retrieval
Commodity Market Information Systems

ENVIRONMENT

Pollution Monitoring
Contamination Studies

Digital Equipment Corporation has now introduced the PDP-8/E. While retaining the fine qualities of the PDP-8 family, the basic architecture has greatly improved. Behind the PDP-8/E is a wealth of engineering know-how and manufacturing capability. This experience has gone into the engineering of the PDP-8/E to create a high quality, low cost machine.

Automated assembly lines are in full operation at DEC. Manufacturing techniques of producing modules on an assembly line basis using the latest process techniques guarantees reliability at low cost.

By utilizing its own digital computer systems, DEC performs hundreds of tests in seconds. To further prove the reliability, DEC tests each component before assembling to a module, then tests each module and finally tests each PDP-8/E processor.

The PDP-8/E is designed for the inexperienced as well as the most sophisticated user. Digital Equipment Corporation provides training as well as maintenance.

PDP-8/E FEATURES

Digital's all new PDP-8/E is the most powerful, most expandable and most versatile 12-bit computer available today. Its low price and high performance makes it the ideal system for a variety of uses, extending all the way from minimal control units to fully expanded general purpose systems. It is fast, compact and easy to interface.

PDP-8/E offers features such as a unique internal bus system called OMNIBUS™, which allows the user to plug memory and processor options into any available slot location: the availability of 256 words of Read-Only or Read/Write memory; a 1.2 microsecond memory cycle time; the use of TTL integrated circuitry with MSI technology; expansion to 32,768 12-bit words of core storage; low-cost mass storage expansion with DECdisk or DECtape; and a space and money saving packaging design.

In addition PDP-8/E provides a standard general-purpose hardware register: an optional program priority interrupt which greatly reduces the response time to service I/O interrupts; FOCAL, Digital's own conversational computer language; and DIBOL, our new COBOL-like business oriented language; and many other new processor options and peripherals.

PDP-8/E Features at a Glance:

- A unique internal bus design called OMNIBUS which eliminates the need for back panel wiring. Processor options can be inserted in any available slot.
- Increased speed-memory cycle time of 1.2 microseconds.
- A new packaging scheme which makes PDP-8/E physically smaller than its predecessor, the PDP-8/I. And, with no predetermined locations needed for options, there is no wasted space in the logic panel.
- A full line of over 60 options and peripherals immediately available.
- More than 1000 programs immediately available to the user.
- Availability of 256 word increments of Read-Only memory and/or Read/Write memory.
- An optionally available advanced program priority interrupt which greatly reduces the response time to an interrupt request.
- A Standard General Purpose register in the basic machine which becomes the MQ register when the EAE option is implemented.

- Six additional Processor IOT instructions which make flag manipulation and interrogation faster and easier.
- A six bit byte swap instruction allowing faster and more convenient character handling.
- TTL integrated circuit modules utilizing MSI technology.
- Over 8500 compatible PDP-8 Family computers in use for sharing programs through Digital's users group, DECUS.
- Low-cost core memory expansion to 32,768 words and low-cost mass storage expansion with DECdisk, DECtape and IBM-compatible magnetic tape.
- Pushdown List option.
- Hardware Bootstrap Loader option.
- A two-module interface for up to 4 teletypes.
- Worldwide, dependable service.
- Program and maintenance training included.
- Fully parallel processor.
- Link feature to facilitate multiple precision arithmetic.
- Full range of turnkey and applications-oriented systems available.
- Over seven years of software development by Digital.
- Expanded hardware multiply/divide.
- Eight auto-index registers.

CONTENTS

PART I PDP-8/E PROGRAMMED DATA PROCESSOR

CHAPTER 1 SYSTEM INTRODUCTION

	Page
GENERAL	1-1
COMPUTER ORGANIZATION	1-2
MAJOR REGISTERS (M830)	1-4
Accumulator (AC)	1-4
Multiplier Quotient (MQ) Register	1-4
Program Counter (PC)	1-4
Central Processor Memory Address (CPMA) Register	1-4
Memory Buffer (MB) Register	1-4
Data Gates and Adders	1-5
REGISTER CONTROLS (M831)	1-5
Link (L)	1-5
Major Register Control Circuits	1-5
Major State Register	1-5
Instruction Register (IR)	1-5
BUS LOADS (M832)	1-6
TIMING GENERATOR	1-6
PROGRAMMERS CONSOLE	1-7
TELETYPE CONTROL (M835)	1-7
PDP-8/E MEMORY SYSTEM (MM8-E)	1-7
The XY Driver & Current Source	1-7
Memory Stack (H220)	1-8
Sense/Inhibit Module (G104)	1-8
MAJOR PROCESSOR STATES	1-8
Fetch (F) State	1-8
Defer (D)	1-9
Execute (E)	1-9
INTERFACING	1-9
DIFFERENCES BETWEEN PDP-8/E AND ITS PREDECESSORS	1-10
TTY Differences	1-11
External I/O Bus	1-11
EAE	1-11
Data Break	1-11
Control Panel	1-12

CHAPTER 2 STANDARD SYSTEM OPERATION

CONTROLS AND INDICATORS 2-1

KEYBOARD OPERATION 2-6

PRINTER OPERATION 2-7

PAPER TAPE PUNCH OPERATION 2-8
Paper Tape Formats 2-9
Paper Tape Loader Programs 2-11

OPERATING PROCEDURES 2-11
Manual Data Storage and Modification 2-12
Power For Manual Operation 2-12
Memory Addressing for Manual Operation 2-12
Manual Data Input To Addressed Memory Location 2-12
Checking the Contents of Any Address in Core 2-12

LOADING DATA UNDER PROGRAM CONTROL 2-12

INITIALIZING THE SYSTEM 2-13

PROGRAM LOADING OPERATION 2-13
Loaders 2-14

READ-IN-MODE (RIM) LOADER 2-14

BINARY (BIN) LOADER 2-17

SYMBOLIC EDITOR 2-20

WRITING A PROGRAM 2-21

GENERATING A PROGRAM TAPE 2-23

SEARCH FEATURE 2-25

ERROR DETECTION 2-26

SUMMARY OF SPECIAL KEYS AND COMMANDS 2-26

PAL III SYMBOLIC ASSEMBLER 2-28

ASSEMBLING A SYMBOLIC PROGRAM 2-29
Program Control 2-33

CHAPTER 3 MEMORY AND PROCESSOR INSTRUCTIONS

MEMORY REFERENCE INSTRUCTIONS 3-3

HOUSEKEEPING INSTRUCTION 3-4

AUGMENTED INSTRUCTIONS 3-5

OPERATE INSTRUCTIONS 3-5
Group 1 3-5
Group 2 3-8
Group 3 3-10

INPUT/OUTPUT TRANSFER (IOT) 3-12
PROGRAM INTERRUPT 3-13
INSTRUCTION SUMMARY 3-15

CHAPTER 4 MEMORY AND PROCESSOR BASIC PROGRAMMING

GENERAL 4-1
MEMORY ADDRESSING 4-1
INDIRECT ADDRESSING 4-5
PROGRAMMING OPERATIONS 4-8
STORING AND LOADING 4-8
ARITHMETIC OPERATIONS 4-10
Two's Complement Arithmetic 4-10
LOGIC OPERATIONS 4-11
Logical AND 4-11
Inclusive OR 4-12
Exclusive OR 4-12
INDEXING OPERATIONS 4-13
CODING A PROGRAM 4-15
Location Assignment 4-15
WRITING SUBROUTINES 4-16
ADDRESS MODIFICATION 4-18
LOOPING A PROGRAM 4-19
AUTO-INDEXING 4-22
PROGRAM DELAYS 4-23
PROGRAM BRANCHING 4-23

MICROPROGRAMMING 4-25
Combining Microinstructions 4-25
Illegal Combinations 4-25
Combining Skip Microinstructions 4-27
OR GROUP—SMA OR SZA OR SNL 4-27
AND GROUP—SPA AND SNA AND SZL 4-27
Group 1 4-28
Group 2 4-28

SYSTEM PROGRAMMING 4-29
PDP-8/E Software Kit 4-30
System Programs 4-30
Monitor Programs 4-30

Disk Monitor System 4-31
Time-Share Monitor 4-32
Editor Program 4-32
Symbolic Paper Tape Editor 4-32
Assembler Programs 4-32
PAL III 4-32
MACRO-8 4-32
8K SABR 4-32
Compiler Programs 4-34
DIBOL Software System 4-34
FORTRAN Compiler 4-35
ALGOL-8 4-35
Interpretive Programs 4-35
FOCAL-8 4-36
BASIC 4-36
Debugger Programs 4-36
DDT-8 4-36
ODT-8 4-37
Loaders 4-37
Read-in-Mode (RIM) Loader 4-37
BINARY (BIN) Loader 4-37
UTILITY Programs 4-37
Data Conversion Programs 4-37
Floating Point Package 4-38

APPLICATION PROGRAMS 4-38

THE DECUS LIBRARY 4-38

CHAPTER 5 PROGRAMMED DATA TRANSFER

GENERAL 5-1

PROGRAMMED DATA TRANSFER V S. DATA BREAK 5-1

PERIPHERAL REQUIREMENTS 5-1

PRINCIPLES OF PROGRAMMED I/O TRANSFERS 5-2
The IOT Instruction 5-2
Flags 5-3
Data Transfers 5-4

PRINCIPLES OF PROGRAM INTERRUPTS 5-5
General 5-5
The Interrupt Subroutine 5-6
The Fag-Testing and Restoration Subroutine 5-6
The Reader Service Routine 5-7
The Punch Service Routine

CHAPTER 6 DATA BREAK

GENERAL 6-1

THE BASIC DATA BREAK SYSTEM 6-1
Current Address (CA) Register 6-2

Word Count (WC) Register 6-2
Data Break Priority 6-2
Data Register 6-2
Data Break Configurations 6-3

ONE-CYCLE DATA BREAK TRANSFERS 6-3
Initial Set-up 6-3
Data Transfer 6-3
Exit 6-7

THREE-CYCLE DATA BREAK TRANSFERS 6-7
Initial Set-up 6-7
Word Count 6-7
Current Address 6-7
Data Transfer 6-10
Exit 6-11

PROGRAMMING EXAMPLE 6-11

BLOCK TRANSFER SUBROUTINE 6-11

PART II PDP-8/E OPTIONS

CHAPTER 7 PDP-8/E OPTIONS

SECTION 1 MECHANICAL EXPANSION OPTIONS 7-1

SYSTEM EXPANDER BOXES 7-1
Type BA8-AA System Expander Box 7-1
Type BA8-AB System Expander Box 7-1
Type BE8-A OMNIBUS Expander 7-1

PANEL OPTIONS 7-1
Type KC8-EC Turn-Key Front Panel 7-1
Type KC8-EB Blank Front Panel 7-1

SECTION 2 COMPUTER INTERNAL OPTIONS 7-1
Type KE8-E Extended Arithmetic Element 7-2
Programming 7-2

MEMORY EQUIPMENT OPTIONS 7-7
KM8-E Memory Extension and Time Share Option 7-7
Memory Extension Description 7-7
Programming 7-8
Time-Share Description 7-13
MP8-E Memory Parity 7-14
Programming 7-15
MW8-E 256-Word Read/Write Memory 7-16
MR8-EA 256-Word Read-Only-Memory 7-17
MR8-EB 1024-Word Read-Only-Memory 7-17
MI8-E Bootstrap Loader 7-17

Real Time Clock Options 7-18
Type DK8-EA Real Time Clock (Line Frequency) 7-18
Programming 7-18
Type DK8-EC Real Time Clock (crystal) 7-18
Type DK8-EP Programmable Real Time Clock 7-19
Programming 7-21
Type KP8-E Power Fail Detect 7-24
Programming 7-25

SECTION 3 OMNIBUS INPUT/OUTPUT EQUIPMENT OPTIONS 7-27
Console Teletype 7-27
Model ASR 33 Teletype 7-27
KL8-E Console Teleprinter Control 7-28
Keyboard/Reader 7-29
Programming 7-29
Teleprinter/Punch 7-31
Programming 7-31

PAPER TAPE READER AND PUNCH OPTIONS 7-33
Type PR8-E Paper Tape Reader 7-33
Programming 7-34
Type PP8-E Paper Tape Punch 7-35
Programming 7-35
Type PC8-E Reader/Punch 7-36
Programming 7-38

CRT DISPLAYS 7-37
Type VC8-E Oscilloscope Display Control 7-37
Programming 7-38
Type VR03A Oscilloscope-Display 7-41
Type VR14 Oscilloscope Display 7-41
Type 374A Light Pen 7-41
Type VA8-E Character Generator 7-41
Type VS8-E Display Control for VT01 Storage Tube Display 7-41
Programming 7-42
Type VT01 Storage Tube Display 7-45
Type H306 JOY Stick Cursor Controller 7-45

X/Y PLOTTER OPTIONS 7-45
Type XY8/E Incremental Plotter Control 7-45
Programming 7-46
Type XY8/EA Digital Incremental Plotter 7-48
Type XY8-EB Digital Incremental Plotter 7-48

LINE PRINTER OPTIONS 7-48
LEG Line Printer 7-48
Programming 7-51

DATA COMMUNICATIONS EQUIPMENT 7-52
DP8-EA and DP8-EB Synchronous Modem Interface 7-52
Programming 7-53
DP8-EP Redundancy Check Option 7-56
Programming 7-57

Interface To Bell 201 Modems 7-58
Interface To Bell 301 and 303 Modems 7-59

CARD READER OPTIONS 7-60
Type CR8-E Card Reader and Control 7-60
Programming 7-61
Type CM8-E Optical Mark Card Reader and Control 7-65
Programming 7-65

LABORATORY PERIPHERALS 7-65
AD8-EA Analog-to-Digital Converter 7-65
Programming 7-66
AM8-E 8-Channel Analog Multiplexer 7-68
Programming 7-69
DR8-EA 12-Channel Buffered Digital I/O 7-71
Programming 7-72
Laboratory Peripheral Panel 7-73

SECTION 4 EXTERNAL BUS INPUT/OUTPUT EQUIPMENT OPTIONS 7-73

EXTERNAL BUS INTERFACE CONTROL OPTIONS 7-73
KE8-E Positive I/O Bus Interface 7-74
BBO8-P General Purpose Interface Unit 7-74
Programming 7-75
KD8-E Data Break Interface 7-76

RANDOM ACCESS DISK DEVICES 7-77
RK8 Disk System 7-77
RKO1 Disk Drive and Control 7-78
RKO8-P Disk Interface Control 7-79
Programming 7-80
DF32-D DEC Disk File and Control and DS32-D DEC Disk File Expander 7-85
Programming 7-86

MAGNETIC TAPE OPTIONS 7-89
DECtape 7-89
TU56 Dual DECtape Transport and TCO8-P DECtape Control 7-95
TU56 Dual DECtape Transport 7-95
TCO8-P DECtape Control 7-97
Programmed Operation 7-99
Programming 7-102
Softwave 7-106

DATA AQUISITION PERIPHERALS 7-108
ADO1A 12-Bit Analog-to-Digital Converter 7-108
Programming 7-110
AFC-8 Low-Level Analog Multiplexer 7-111
Programming 7-113
Type AFO4 Guarded Scanning Integrating Digital Voltmeter 7-114
Programming 7-115
Frequency and Period Measurement Options for AFO4A 7-118
Additional AFO4A Options 7-119

AA50 Digital-to-Analog Converter 7-120
Programming 7-119
AA05A/AA07 Digital-to-Analog Converter and Control 7-121
Programming 7-122
Universal Digital Controller (UDC) 7-122
Programming 7-124

PART III INTERFACING & INSTALLATION

CHAPTER 8 DIGITAL LOGIC CIRCUITS

INTRODUCTION 8-1

LOGIC SYMBOLS 8-1
State Indicator 8-1
Table of Combinations 8-3
One-Shot Functions 8-4
Schmitt Trigger 8-4
General Logic Symbols 8-5
Amplifier 8-5
Time Delay 8-6

CHAPTER 9 THE OMNIBUS INTERFACING

INTRODUCTION 9-1

SECTION 1 OMNIBUS DESCRIPTION 9-3

BUS STRUCTURE 9-3

BUS SPECIFICATIONS 9-4

SYSTEMS CONFIGURATION 9-4

RELATIONSHIP OF THE EXTERNAL BUS TO THE OMNIBUS 9-6

OMNIBUS SIGNALS 9-6

DATA TRANSFER, ADDRESSING, AND DATA CONTROL LINES 9-19
Data Paths 9-20
Device Addressing 9-21
Data Break Addressing 9-22
Control Signals 9-22

DEVICE CODES 9-22

SECTION 2 INTERFACE DESIGN 9-23

DESIGNING AN INTERFACE CONTROL MODULE 9-23
Board Layout 9-24
Etched Circuit Layout and Construction Rules 9-25
Designing Basic Interface Control Circuits 9-26
General Cable Rules and Suggestions 9-28

DEC Supplied Interface Cables 9-28
Cabling Rules

INTERFACE TIMING CRITERIA 9-30
General Timing Rules 9-30
Interrupt Timing 9-30
Timing Example 9-30
Timing Requirements for Data Break Facilities 9-32

GENERAL PROPAGATION DELAY GUIDELINES 9-33
2-input NAND Gate Delay 9-33
Flip-Flop Propagation Delays 9-33
J-K Flip-Flops 9-34
One-Shot Delays 9-34

MAXIMUM OPERATING FREQUENCY 9-35

LOADING RULES 9-37
Device Selection Inputs 9-37
Skip and Interrupt Request Lines 9-37
Electrical Considerations of Driving a Line 9-37

GROUNDING 9-37

TESTING TECHNIQUES 9-38
Initial Checkout 9-38
System Test 9-38
Final Testing 9-38

PROGRAMMING RULES 9-38

APPLICATION EXAMPLE 9-39

CHAPTER 10 EXTERNAL BUS INTERFACING

GENERAL 10-1

THE NATURE OF THE EXTERNAL BUS 10-1

EXTERNAL BUS SIGNALS 10-2

APPLICATION 10-7
Programmed I/O Transfer 10-7
Program Interrupt Transfers 10-12
Data Break Transfers 10-12

INTERFACING TECHNIQUES 10-16
PDP-8/I and 8L-Type Peripherals 10-17
Customer Peripherals 10-19
DEC Logic Module Interfacing 10-19
Customer Designed Interfaces 10-27
Restrictions and Criteria 10-29
Cooling
Signal Terminating
Timing Criteria

CABLING RULES AND SUGGESTIONS 10-30

M-SERIES MODULE SUMMARY 10-32

CHAPTER 11 INSTALLATION AND PLANNING

SPACE REQUIREMENTS 11-1

ENVIRONMENTAL REQUIREMENTS 11-5

INSTALLATION PROCEDURE 11-6

SYSTEM CONFIGURATIONS 11-12

PART IV DEC SERVICES

CHAPTER 12 EQUIPMENT SUPPORT SERVICES

INTRODUCTION 12-1

SECTION 1

MAINTENANCE AND SERVICE OPTIONS 12-1
- Service Contracts 12-1
- Service Contract Options 12-1
- Eligibility for Service Contract Coverage 12-2
- Depot Repair 12-2

FIELD INSTALLATION OF ADDITIONAL DEC OPTIONS 12-2

SECTION 2

THE DEC TRAINING PROGRAM 12-2
- Course Offerings 12-2
- Course Objectives 12-3
- Programming 12-3
- Hardware Familiarization 12-3
- Systems Maintenance 12-3

COURSE DESCRIPTIONS 12-3
- Basic Programming 12-3
- PDP-8 Family Programming (Paper-Tape System) 12-4
- PDP-8 Family Programming (DECtape/Disk System) 12-4
- PDP-8/E Hardware Familiarization 12-4

SECTION 3

ECO SERVICE 12-4

APPENDIX A PROGRAM ABSTRACTS

FAMILY-OF-8 PROGRAMS A-1

SYSTEM PROGRAMS A-1
Monitor Programs A-1
Disk Monitor System A-2
Editor Programs A-3
Assembler Programs A-3
Compiler Programs A-4
Interpretive Programs A-6
Debugging Programs A-7
Loader Programs A-8
Utility Programs A-8

APPENDIX B TABLE OF INSTRUCTIONS B-1

APPENDIX C TABLE OF CODES C-1

APPENDIX D PERFORATED-TAPE LOADER SEQUENCES D-1

APPENDIX E LINE PRINTER CHARACTER CODES E-1

APPENDIX F SCALES OF NOTATION F-1

APPENDIX G POWERS OF TWO G-1

APPENDIX H OCTAL-TO-DECIMAL CONVERSION H-1

PDP-8/E Programmed Data Processor
(Table Model)

part 1

PDP-8/E PROGRAMMED DATA PROCESSOR

CHAPTER 1

SYSTEM INTRODUCTION

GENERAL

The development of the PDP-8/E is the successful culmination of many years of computer design research—a process that has enabled Digital Equipment Corporation to provide better computers at the lowest possible price.

The PDP-8/E is specially designed as a general purpose computer. It is fast, compact, inexpensive, and easy to interface. The PDP-8/E is designed to meet the needs of the average user and is capable of modular expansion to accommodate most individual requirements for a user's specific applications.

The PDP-8/E basic processor is a single-address, fixed word length, parallel-transfer computer using 12-bit, 2's complement arithmetic. The cycle time of the 4096-word random address magnetic core memory is 1.2 microseconds for fetch and defer cycles without autoindex; and 1.4 microseconds for all other cycles. Standard features include indirect addressing and facilities for instruction skip and program interrupt as a function of the input/output device condition.

Five 12-bit registers are used to control computer operations, address memory, operate on data and store data. A Programmer's console provides switches to allow addressing and loading memory and indicators to observe the results. The PDP-8/E may also be programmed using the console Teletype with a reader/punch facility. Thus, programs can be loaded into memory using the switches on the Programmer's console, the Teletype keyboard, or the paper tape reader. Processor operation includes addressing memory, storing data, retrieving data, receiving and transmitting data and mathematical computations.

The 1.2/1.4-microsecond cycle time of the machine provides a computation rate of 385,000 additions per second. Each addition requires 2.6 microseconds (with one number in the accumulator) and subtraction requires 5.0 microseconds (with the subtrahend in the accumulator). Multiplication is performed in 256.5 microseconds or less by a subroutine that operates on two signed 12-bit numbers to produce a 24-bit product, leaving the 12 most significant bits in the accumulator. Division of two signed 12-bit numbers is performed in 342.4 microseconds or less by a subroutine that produces a 12-bit quotient in the accumulator and a 12-bit remainder in core memory. Similar signed multiplication and division operations are performed in approximately 40 microseconds, utilizing the optional Extended Arithmetic Element.

The flexible, high-capacity input/output capabilities of the computer allow it to operate a large variety of peripheral machines. Besides the standard keyboard and paper-tape punch and reader equipment, these computers are capable of operating in conjunction with a number of optional devices (such as high-speed perforated-tape punch and reader

equipment, card reader equipment, line printers, analog-to-digital converters, cathode ray tube (CRT) displays, magnetic tape equipment, a 32,764-word random-access disk file, a 262,112-word random-access disk file, etc.).

The PDP-8/E system is completely self-contained, and requires no special power sources or environmental conditions. A single source of 115V or 230V at 47 to 63 Hz, single-phase power is required. Internal power supplies produce the necessary operating voltages for the system.

The user has a choice of two basic configurations. Table Top or Rack Mountable. Standard DEC cabinets are available to accommodate those users desiring many peripherals. The Table Top version is a convenient approach for those desiring to use the processor in a small area, such as an office.

COMPUTER ORGANIZATION

The PDP-8/E system consists of a central processor, core memory, and input/output equipment facilities; all of which interrelate by means of a common bus called "OMNIBUS."

All arithmetic, logic, and system control operations are performed by the central processor. Information storage and retrieval operations are performed by the core memory. The memory is continuously cycling, automatically performing a read and write operation during each computer cycle. Input and output address and data buffering of the core memory are performed by registers in the central processor, and the operation of the core memory is under control of timing signals produced by the central processor. Because of the close relationship of operations performed by the central processor and the core memory, both are described in this chapter.

Central processor interface circuits provide bussed connections to a variety of peripheral input/output equipment. Each input/output device is responsible for detecting its own select code and for providing all required input or output gating. Individually programmed data transfers between the central processor and peripheral equipment take place through the central processor accumulator. Data break transfers can be initiated by peripheral equipment, rather than under program control, through the data break facilities. Standard features of the computer allow peripheral equipment to perform certain control functions, i.e., instruction skipping and a transfer of program control initiated by a program interrupt.

Standard equipment provided with each system includes a console teleprinter control, which drives and controls a Teletype Model 33 Automatic Send-Receive (ASR). The ASR set is a standard machine operating from serial 11-unit code characters at a rate of 10 characters per second. The ASR provides a means of supplying data to the computer from keyboard or perforated tape; and supplies data as output from the computer in the form of typed copy, or typed copy and perforated tape. The Teletype control serves as a serial-to-parallel converter for teletype inputs to the computer and serves as a parallel-to-serial converter for computer

output signals to the Teletype unit. The Teletype and other input/output equipment options are discussed in Chapter 7 of this handbook.

The basic system is illustrated in Figure 1-1. It contains nine PDP-8/E FLIP-CHIP modules called: Major register (M830); Register Control (M831); Bus-Loads (M832); Timing Generator (M833); Panel type KE8-EA; Teletype control (M856); XY Driver and Current Source (G227); Memory Stack (H220); and Sense/Inhibit (G104). (The last three modules are memory system modules.)

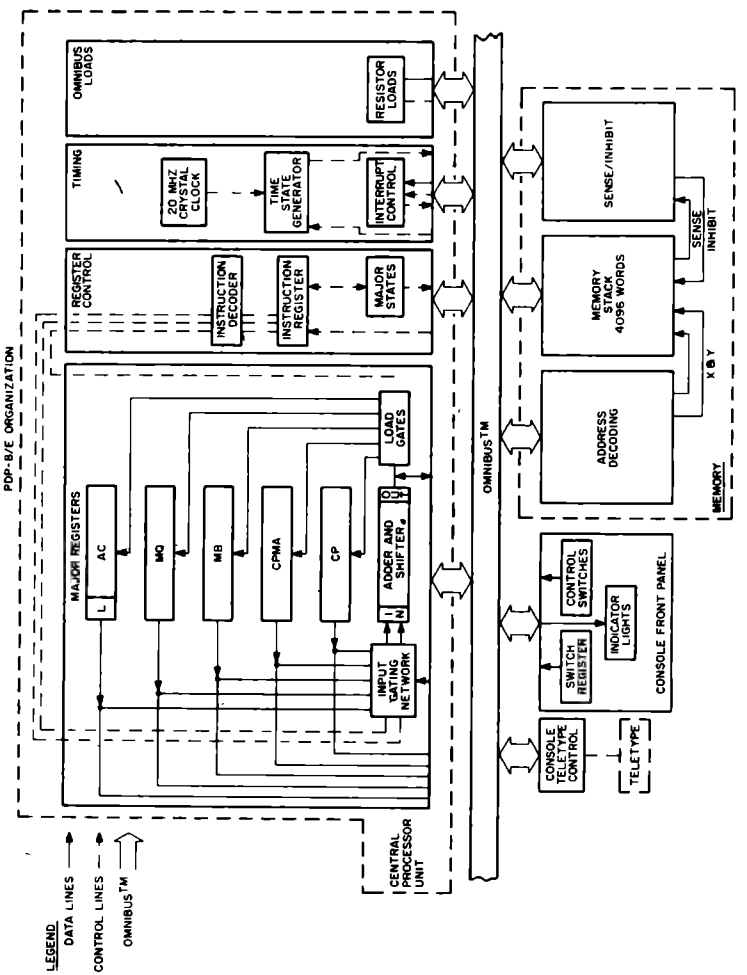


Figure 1-1 PDP-8/E Basic System Block Diagram

MAJOR REGISTERS (M830)

In order to store, retrieve, control, and modify information and to perform the required logical, arithmetic, and data processing operations, the core memory and the central processor employ the logic complement and major registers shown in Figure 1-1 and described in the following paragraphs.

Accumulator (AC)

The AC is a 12-bit register in which arithmetic and logic operations are performed. Under program control the AC can be cleared or complemented, or its contents can be rotated right or left. The contents of the memory buffer register can be added to the contents of the AC (via the adder circuit), and the result stored in the AC. The contents of both of these registers can be combined by the logical AND operation with the result remaining in the AC. The inclusive OR may be performed between the AC and the switch register (on the programmer's console), and the result left in the AC. The AC also serves as an input/output register; all programmed information transfers between the core memory and an I/O device are passed through the AC to data lines located on the OMNIBUS.

Multiplier Quotient (MQ) Register

The MQ is a 12-bit bidirectional shift register that acts as an extension of the AC during EAE operations. The MQ contains the multiplier at the beginning of a multiplication and the least significant half of the product at the conclusion. The MQ contains the least significant half of the dividend at the start of a division and the quotient at the end. The MQ contains the least significant part of a number during a shift or a normalize operation. The MQ is also available as a temporary storage register adding additional capability and flexibility for the programmer.

Program Counter (PC)

The PC is a 12-bit register that is used to control the program sequence; that is, the order in which instructions are performed is determined by the PC. The PC contains the address of the core memory location from which the next instruction is taken. Information enters the PC from the core memory via the Memory Buffer and from the Memory Address Register. Information in the PC is transferred into the Memory Address register to determine the core memory address from which each instruction is taken. Incrementing the contents of the PC establishes the successive program core memory locations and provides skipping of an Instruction based upon a programmed test of information or conditions.

Central Processor Memory Address (CPMA) Register

The CPMA register is a 12-bit register that contains the address in core memory that is currently selected for reading or writing. Therefore, all 4096 words of core memory can be addressed directly by the CPMA. Data can be transferred into the CPMA from the Memory Buffer, from the Program Counter and from the switch register on the operator's console. Memory Addressing is also accomplished by each data break interface (refer to Chapter 6). This register is never cleared. New information is always jam transferred in and the original content is lost.

Memory Buffer (MB) Register

The MB register is a 12-bit register that is used for all information

transfers between the central processor registers and the core memory. Information can be transferred and temporarily held in the MB from the AC or PC. Also, the MB can simultaneously be loaded and incremented by one before being read back into memory. Information can be loaded into the MB from an I/O device during a data break or from core memory. Information is read from a memory location in 0.6 microseconds and rewritten in the same location in another 0.6 microsecond of a single 1.2-microsecond duration memory cycle. Many machine cycles require modification of memory data. In such cycles, an extra 0.2 microsecond is inserted between read and write.

Data Gates and Adders
The Major Registers module also contains the gating necessary to move data from one register to another. At the heart of the data gating is a 12-bit parallel adder. Information from a register is gated to the adder inputs. The output of the adder is applied to a set of shift gates. The output of the shift gates serves as data input to all of the major registers.

REGISTER CONTROLS (M831)
The Register Control module contains the Link, the Major Register Control circuits, the Major States register, the Instruction register, and the necessary control circuits for the Major Sates register and the Instruction register.

Link (L)
The Link is a 1-bit register that is used to extend the arithmetic facilities of the AC. It is used as the carry register for 2s complement arithmetic. Overflow into the L from the AC can be checked by the program to greatly simplify and speed up single and multiple-precision arithmetic routine. Under program control, the L may be either cleared, complemented, or rotated as part of the AC.

Major Register Control Circuits
The Major Register Control Circuits enable the adder input and shift gates of the Major Register module. They also gate time pulses to cause loading of the appropriate major register.

Major State Register
The Major State register is the control for the three major states of the PDP-8/E. Conditional inputs, consisting of processor instructions combined with the output of the Major State register, determine which of the three major states (FETCH, DEFER, or EXECUTE) the processor is about to enter. Each of the major state signals, when asserted, is used to enable the corresponding register control circuitry.

Instruction Register (IR)
The IR is a 3-bit register that contains the operation code of the instruction currently being performed by the machine. The three most significant bits of the current instruction are loaded into the IR from the memory during a fetch cycle. The contents of the IR are decoded to produce the eight basic instructions and affect the cycles and states entered during each step of the program.

BUS LOADS (M832)

The Bus Loads module contains all of the necessary load resistors required to maintain a high inactive level for each of the busses in the system. There are basically seven groups of signals that the Bus Loads module services. These are: 1. Memory Address (MA); 2. Memory Data (MD); 3. Data; 4. I/O Control; 5. Break Control; 6. Timing; and 7. Miscellaneous Signals. Most lines are considered by the system to be inactive (voltage level high) until the line level is pulled to ground by some component connecting to the corresponding signal line.

TIMING GENERATOR (M833)

The Timing Generator module contains the time pulse generator, Interrupt Control circuits, the Processor IOT Decoder, and miscellaneous control circuits.

The time pulse generator provides the timing pulses that determine the computer cycle time and are used to initiate sequential time-synchronized gating operations. Pulses that reset registers and control circuits during power turn-on and turn-off operations are produced by the power clear pulse generator. Several of these pulses are available for peripheral device control to be utilized with devices using programmed or data break information transfers.

Four time states, TS1 through TS4, are provided by the time pulse generator. In addition, four time pulses are generated for use as gating pulses throughout the system. Each time pulse overlaps the end of one time state into the beginning of the next time state (refer to Figure 1-2). Memory timing is also provided by the time pulse generator.

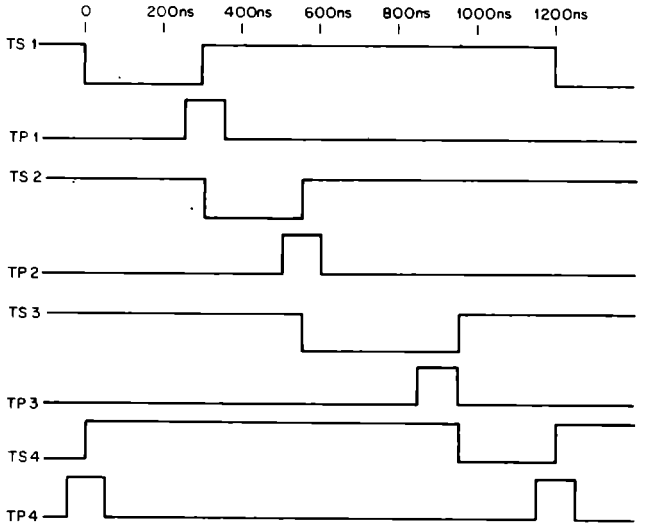


Figure 1-2 Timing Diagram

The Interrupt Control circuits comprise the major portion of the Interrupt System. The circuitry responds whenever an INTERRUPT REQUEST signal is received from an interface controller module.

The Processor IOT Decoder decodes the last 9 memory data bits and determines the type of IOT instruction that is to be performed.

PROGRAMMER'S CONSOLE (KE8-EA)
The Programmer's Console contains a convenient array of controls and indicators that are used specifically for operation and maintenance. The Console has been configured to achieve convenient control of the system. Through switches and keys on the Console, the operator can STOP, START, EXAMINE, MODIFY, or CONTINUE a program. The indicators, when properly selected, display the machine status and contents of major registers.

A lighted indicator denotes the presence of a Binary 1 in a specific register bit position or control flip-flop.

The Programmer's Console contains a 12-bit switch register, ten control switches, and an indicator selector switch capable of selecting seven individual major states and status registers to be displayed on a 28-lamp indicator panel.

TELETYPE CONTROL (M835)
The Teletype Control Module contains the necessary receive and transmit circuitry along with the control circuitry to interface the ASR 33 Teletype terminal with the processor.

PDP-8/E MEMORY SYSTEM (MM8-E)
The basic PDP-8/E memory system (MM8-E) is a 4096 word, 12-bit random access core memory that performs all normal functions of data storage and retrieval. The same basic 4K memory, consisting of 3 quad modules, can be used as an extended memory to increase the memory capacity up to the addressing capability (32K) of the PDP-8/E.

Memory location 0 is used to store the contents of the PC following a program interrupt, and location 1 is used to store the first instruction to be executed following a program interrupt. When a program interrupt occurs, the contents of the PC are stored in location 0 and program control is transferred to location 1 automatically. Core memory locations 10 (octal) through 17 (octal) are used for auto-indexing. All other locations can be used for the storage of either instructions or data.

The memory system contains circuits such as read/write switches, address decoders, inhibit drivers, and sense amplifiers. These circuits perform the electrical conversions necessary to transfer information to or from the core array. They perform no arithmetic or logic operations upon the data.

The XY Driver & Current Source (G227)
This PDP-8/E module contains the circuitry required to decode the address lines and drive the XY wires of a 4096 word core memory (i.e., address decoding, selection switches, XY current sources, stack discharge switch, and power on/off write protection). The XY currents are

controlled remotely by a control on the Sense/Inhibit board. The XY Driver and Current Source module requires no adjustments. The same module is used also in the memory parity option.

Memory Stack (H220)

The memory cores are mounted on a G619 Planar Stack Board. The whole module assembly is the H220 Stack. It contains 4096 words of 12-bit core memory and the X-axis and Y axis diode selection matrix. It also includes a resistor/thermistor combination that supplies temperature information to the XY current control. The core memory is a 3D/3-wire memory with center tapped sense/inhibit wire. This module has no connections from and to the OMNIBUS. The same stack is also used in the memory parity option.

Sense/Inhibit Module (G104)

The Sense/Inhibit module (G104) is a PDP-8/E module containing the sense amplifiers, memory register, and the inhibit drivers for a word length of 12 bits. It also includes the slice control and the −6V supply for the sense amplifiers, the current control for the XY current source, control logic for the strobe and clear, and the field select, which is used in the Sense/Inhibit as well as in the XY Driver. Three jumper connections determine the field. Slice level, strobe delay, and XY current can be selected within four discrete steps by appropriate jumper connections (two per axis). With a given stack the proper combination is known and the jumper connection can be selected. Adjustments in a system are, therefore, eliminated.

The memory parity option, consisting of another 3 modules, adds all the circuitry necessary to read, write, and store the parity for 32K of memory. Additional memory options are a 256-word Read Only Memory, a 1024-word Read Only Memory, a 256-word Read/Write Memory, and a Bootstrap Loader Read Only Memory. Refer to Chapter 7 for the description of each memory option.

MAJOR PROCESSOR STATES

The PDP-8/E utilizes three processor states to execute programmed instructions. To accomplish this, a major state generator is used to establish one state for each computer timing cycle. The major processor states are: FETCH, DEFER and EXECUTE. FETCH, DEFER and EXECUTE states determine and execute instructions.

Fetch (F) State

The computer enters the FETCH state to obtain a 12-bit instruction word. At the start of FETCH cycle, the contents of the PC are loaded into the CPMA, giving the first memory address and starting the memory cycle.

At the start of the FETCH cycle, the computer obtains the contents of an addressed memory location and places the 12 bits on the Memory Data lines of the OMNIBUS. The contents of these lines are decoded to determine the kind of instruction the processor must next perform. Once the processor decides the kind of instruction it must do, it then begins performing the instruction, entering a DEFER, or an EXECUTE state as required. When the instruction has been completely performed, the computer again enters the FETCH state to obtain the next instruction.

Assuming the computer is a fully automatic, running condition; that is, the computer is continuously functioning to FETCH and/or EXECUTE instructions, the PC register's contents are transferred to the MA register at TP4 time, initiating the FETCH state. When this is accomplished, the MA is simultaneously incremented by one and the result is transferred to the PC register. Sometime during TS2, the memory is strobed. The contents of the memory appear at the output of the sense amplifiers and are transferred to the MD lines. The contents of the first three bits of the memory data are transferred to the IR. This completes the activity during time TS2 of the FETCH state.

If the instruction is a multicycle (2 or 3) instruction, the memory address is computed, but no further action takes place until the Defer or Execute cycle. If, however, the instruction is a single cycle instruction (such as IOT, OPR, or JMP and bit 3 = 0) the instruction is carried out immediately. The Major State register gating causes the computer to enter the appropriate major state at the end of the cycle.

Defer (D) State
At the end of a FETCH cycle, the current instruction is directed to DEFER whenever indirect addressing is required.

The DEFER state is entered if a binary 1 is present in bit 3 of a memory reference instruction. This state causes the central processor to obtain the full 12-bit address of the operand from an address in either the current page or page zero, as specified by bits 4 through 11 of the instruction. This process of address deferring is called indirect addressing, because access to the operand is addressed indirectly, or deferred, to another memory location.

Execute (E) State
The EXECUTE state is entered for all memory reference instructions except JMP. During an AND, 2s complement add, or increment and skip if zero instruction, the contents of the core memory location specified by the address portion of the instruction are read into the MB and the operation specified by the Instruction Register is performed. During a deposit and clear accumulator instruction, the contents of the AC are transferred into the MB and stored in core memory at the address specified in the instruction. During a jump to a subroutine instruction, the EXECUTE state occurs to write the contents of the PC into the core memory location as designated by the instruction and to transfer this address +1 into the PC to bring about a change in program control.

In addition to the three major states described, the processor responds to other functions such as Data Break. During this time period, FETCH, DEFER and EXECUTE states are held inactive.

INTERFACING

The PDP-8/E offers two approaches to interfacing with peripheral equipment. The OMNIBUS is an internal Input/Output Bus on which all I/O data and control signals are transferred. A variety of peripherals can interconnect through a peripheral control module from this bus. The OMNIBUS has eliminated wires by providing an etched circuit board on

which connectors are mounted. For the convenience of the user, each pin assignment on one connector is identical to the next connector, allowing a module to be placed anywhere on the bus. The PDP-8/E options provide a complete line of peripherals used in most computer operations. However, for those requirements that are not covered by options, Chapter 9 provides the information needed so that a user can build his own interface. Special interface modules can also be constructed by DEC.

An External Bus is the second approach to interfacing to peripherals. The External Bus connects to the OMNIBUS and provides an extension to the bus system for users who already possess or desire to use PDP-8/I or PDP-8/L compatible peripherals. The interfacing details are provided in Chapter 10 of this handbook.

Three types of data transfer systems are available to the user. One system is the straight Input/Output transfer, which is the simplest and most direct type of transfer. The Program Interrupt system is a type of data transfer system useful for installations having more than one peripheral. For installations using extremely fast transfer rates, the data break system (sometimes called Direct Memory Access, or DMA) is available (refer to Chapter 6 for details on the Data Break system and Chapter 5 for details on programmed transfers).

DIFFERENCES BETWEEN PDP-8/E AND ITS PREDECESSORS

As a new computer is developed, differences between it and its predecessors inevitably result. In many cases these differences are either benign or beneficial. All known differences are listed below in order to give users of earlier machines a concise summary.

Instruction Differences

NEW INSTRUCTION	OCTAL	PREVIOUS FUNCTIONS
Byte Swap (BSW)	7002	Rotate 2, no direction (no operation)
Swap MQ and AC (SWP)	7521	MQL MQA (worked as a SWP in KE8-I, but not documented)
Reserved for future expansion	7014	RAR RAL*
Reserved for future expansion	7016	RTR RTL*
MQ instructions	74X1	Only available with EAE on previous machines; otherwise produced a NOP.
Skip if interrupt on (SKON)	6000	No operation
Skip on interrupt request (SRQ)	6003	(ION)
Get flags (GTF)	6004	No operation, or ADC in PDP-8 with 189 A/D Converter
Restore flags (RTF)	6005	(ION) (ORed with ADC)
Skip if Greater Than (SGT)	6006	(IOF) (ORed with ADC)
Clear all flags (CAF)	6007	(ION) (ORed with ADC)

* These instructions produced predictable but undocumented results in PDP-8/I and PDP-8/L. They may have been used by some programmers to load the constants 3776 and 5775 into the AC. In the PDP-8/E, these codes are specifically reserved for future expansion, and should not be executed.

TTY Differences

The console TTY uses the same IOT's as in earlier machines, but also has added IOTs to enable or disable TTY flags onto the interrupt bus. Also included are additional IOTs to clear keyboard flags without advancing the reader and an IOT to set the printer flag. The skip IOT's can no longer be microprogrammed with the other IOTs of the same device code. This should impose no constraint on the user, since skips are generally not combined with other IOT's. Reader Run is no longer set by INITIALIZE. Hence any routines using the reader must begin with a KCC instruction.

External I/O Bus

In general, the signals and functions at the External I/O Bus Interface are the same as for previous machines. Users who constructed peripherals using previous editions of the Small Computer Handbook as a guide may expect their peripherals to work on the PDP-8/E. (Please note, however, that PDP-8/E is equipped only with a positive bus; and a DW08A bus converter is necessary to interface to older, negative bus equipment.)

The BAC lines at the External I/O Bus interface are merely the buffered DATA bits of the OMNIBUS. Since the DATA lines are used for bi-directional transfer, any input of data at the External I/O Bus interface will cause an immediate change at the BAC outputs. Simultaneous input and output transfers in the same IOP should be checked. In such situations, the register in the peripheral must be edge-triggered.

Also, at the conclusion of the IOP dialogue, the DATA bus is used for updating the PC, and then for determining break or API priority. Users can no longer rely on the BAC lines being available until the end of the major state. However, the IOP width and separation may be adjusted if desired, to accommodate slow I/O devices. External IOTs are faster in all cases except for IOTs ending in 7.

EAE

The Extended Arithmetic Element has been redesigned, and several powerful features have been added. Previous EAE users will have to recode their programs. It is a wise move to recode the EAE programs anyway, in order to make use of the new SAM, DCM, DAD, DST, DPIC and DPSZ instructions.

Data Break

The time required to access the Data Break system has been greatly reduced. Maximum benefit can be obtained on machines without EAE, and with only internal options (or options which are not activated while Data Break is in use). An added feature, ADM, allows the user to add an input word to the contents of a memory location. An internal mul-

tiplexing scheme allows the use of several (12 max.) external and/or internal break devices.

Control Panel

The control panel of PDP-8/E differs from panels of its predecessors as follows:

1. Only the MA (and EMA) and the RUN status are permanently displayed. All other registers are selected by a rotary switch.
2. The machine stops after TP4. Thus the MA lights indicate the next address to be accessed. Similarly, the Major State indicators show the next major state to be executed.
3. Extended field information is loaded via SR6-11 and the EXTD ADDR LOAD switch.
4. Operation of the ADDR LOAD switch places the CP in the FETCH state.
5. Programs are started by operating and releasing the CLEAR key, then operating and releasing the CONT key.

CHAPTER 2

STANDARD SYSTEM OPERATION

GENERAL

The PDP-8/E Computer allows the operator to manually program the machine using the switch register located on the Programmer's Console or use the more automatic process with the ASR 33 Teletype Console which contains the Tape Reader/Punch combination as well as the standard Teletype keyboard. This chapter defines the operation of communicating with the processor in both the manual and program control modes.

The user should be thoroughly acquainted with the content of chapters 3 and 4 before operating the system.

CONTROLS AND INDICATORS

The controls and indicators on the Programmer's Console provide manual control and indicate the program conditions of the PDP-8/E. Controls on the Programmer's Console provide the operator with the hardware to start, stop, examine, modify, or continue a program. The indicators on the console provide a visual indication of the machine status and current program, the contents of the major registers, and the condition of the control flip-flops. A lighted indicator denotes the presence of a binary 1 in a specific register bit position or control flip-flop. Table 2-1 lists the functions of controls and indicators. The controls are divided into two groups; switches and keys. Keys are momentary, or spring-return, switches. Figure 2-1 illustrates the console. Controls and indicators of the standard Model 33 ASR Teletype unit are shown in Figure 2-2 and their functions are described in Table 2-2.

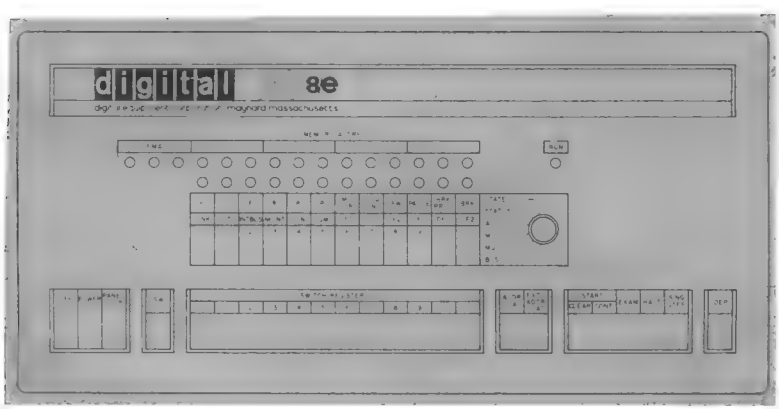


Figure 2-1 Programmer's Console

Table 2-1

Programmer's Console Control and Indicator Functions

CONTROL OR INDICATOR	FUNCTION
Off/Power/Panel Lock	This is a key operated switch. In the counter-clockwise, or OFF, position, the switch disconnects all primary power to the machine. In the POWER, or straight up position, it enables all manual controls and applies primary computer power. In the PANEL LOCK or clockwise position, it disables all keys and switches with the exception of the switch register and mode switch. In this position, a running program is protected from inadvertent switch operation and all panel indicators except the RUN light are turned off.
SW	When this switch is up, the line on the OMNIBUS line called SW is high; when the lever is down, the line is low. This switch is used by special peripheral controls, such as the Bootstrap Loader.
Switch Register Switches (SR)	These 12 switches provide a means of communication between operator and machine. They allow a 12-bit word to be input. When the switch is up it designates a binary 1 to the machine; switch down is a 0 (zero). These switches are used during manual functions or under program control.
Load Address Key (ADDR LOAD)	This key loads the contents of the Switch Register into the CPMA and forces Fetch to be set (no Major States while the Load Address Key is depressed).
Extended Address Load Key (EXTD ADDR LOAD)	This switch loads the contents of SR6-11 into the Data Field and Instruction Field registers of the Memory Extension Control. SR9-11 goes to Data Field 0-2. SR6-8 goes to Instruction Field 0-2.
Clear Key (CLEAR)	This key issues an Initialize Pulse, clearing the AC, LINK, Interrupt system, and I/O Flags.

Table 2-1 (Cont.)

CONTROL OR INDICATOR	FUNCTION
Continue Key (CONT)	This key resumes the computer program by issuing a Memory Start and setting the Run Flip-Flop. The word stored at the address currently held by the CPMA is taken as the first instruction.
Examine Key (EXAM)	Puts the contents of core memory at the address specified by the contents of the CPMA into the MB. Then the contents of the PC and CPMA are incremented by one to allow examination of the contents of sequential core memory addresses by repeating the operation of the examine switch.
Halt Switch (HALT)	This switch clears the Run flip-flop and causes the machine to stop at TS1 of the next Fetch cycle. This switch is also used for single instruction stepping.
Single Step Switch (SING STEP)	This switch clears the Run flip-flop and causes the machine to stop at TS1 of the next cycle. Thereafter, repeated depressing of the continue key steps the program one cycle at a time, so that the contents of registers can be observed in each state.
Deposit Key (DEP)	Loads the contents of the SR into the MB and core memory at the address given by the current contents of the CPMA. Then the contents of the PC and CPMA are incremented by one. This allows storing of information in sequential memory address by repeated operation of the deposit switch.
Indicator Selector Switch	This is a six-position rotary switch, used to select a register for display. The six positions are as follows: 1. STATE — Indicates an individual function for each bit; Bit 0—Fetch 1—Defer 2—Execute 3—IR 0 4—IR 1 5—IR 2

Table 2-1 (Cont.)

CONTROL OR INDICATOR	FUNCTION
	6—MD DIR 7—Data Control 8—SW 9—Pause 10—Break in Prog 11—Break
	2. STATUS — Indicates an individual function for each bit; Bit 0—Link 1—Greater Than Flag 2—Interrupt Bus 3—No Interrupt Allowed 4—Interrupt On 5—User Mode 6—Instruction Field 0 7—Instruction Field 1 8—Instruction Field 2 9—Data Field 0 10—Data Field 1 11—Data Field 2
Indicator Selector Switch	3. AC — Indicates bits 0-11 of the accumulator at TS1.
	4. MD—Indicates Information just written or rewritten into memory.
	5. MQ—Indicates contents of MQ register during TS1.
	6. BUS—Indicates bits 0-11 of the DATA Lines.
Memory Address	Indicates the contents of the memory address which will be accessed next.
EMA	Indicates which Extended Memory field is being accessed.
Run Light	When lit means machine's timing is enabled and capable of executing instructions.

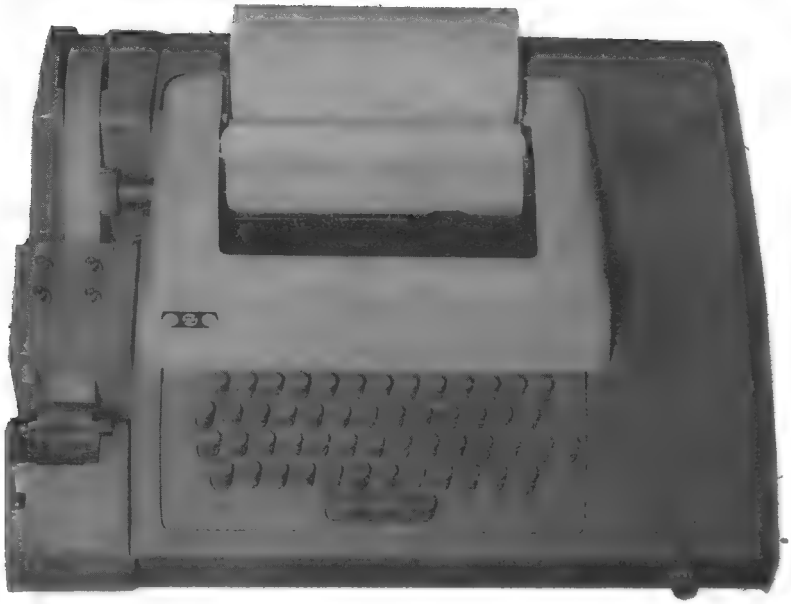


Figure 2-2 Teletype Model ASR33 Console

Table 2-2

ASR33 Teletype Controls and Indicators

CONTROL OR INDICATOR	FUNCTION
REL Pushbutton	Disengages the tape in the punch to allow tape removal or tape loading.
B SP Pushbutton	Backspaces the tape in the punch by one space, allowing manual correction or rubout of the character just punched.
OFF/ON Pushbuttons	Controls use of the tape punch with operation of the Teletype keyboard/printer.
START/STOP/FREE Switch	Controls use of the tape reader with operation of the Teletype. In the FREE position the reader is disengaged, permitting the paper tape to be manually moved within the reader without necessarily reloading or unloading it. In the STOP position the reader mechanism is

Table 2-2 (Cont.)

CONTROL OR INDICATOR	FUNCTION
	engaged but de-energized. In the START position the reader is engaged and operated under program control. The tape may be loaded or unloaded in either the FREE or STOP positions.
Keyboard	Provides a means of printing on paper when used as a typewriter and punching tape when the punch ON pushbutton is pressed; also provides a means of supplying input data to the computer when the LINE/OFF/LOCAL switch is in the LINE position.
LINE/OFF/LOCAL Switch	Controls application of primary power to the Teletype and data connection to the processor. In the LINE position, the Teletype is energized and connected as a computer I/O device. In the OFF position, the Teletype is de-energized. In the LOCAL position, the Teletype is energized for off-line operation, and signal connections to the processor are disconnected. Both LINE and LOCAL use of the Teletype require that the computer be energized through the POWER switch.

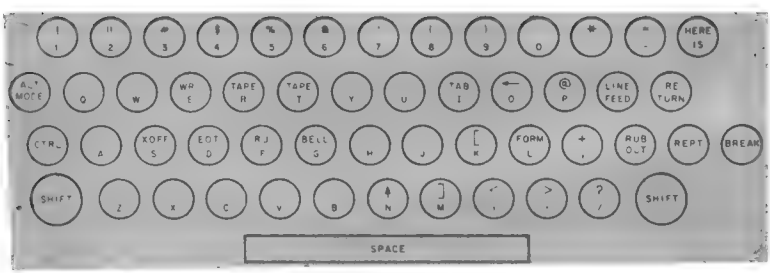


Figure 2-3 Teletype Keyboard

KEYBOARD OPERATION

The Teletype keyboard shown in Figure 2-3 is similar to a typewriter keyboard, except that some nonprinting characters are included as upper case elements. For typing characters or symbols, such as $, %, #, which appear on the upper portion of numeric keys and certain

alphabetic keys, the SHIFT key is held depressed while the desired key is operated.

Designations for certain nonprinting functions are shown on the upper part of some alphabetic keys. By holding the CTRL (control) key depressed and then depressing the desired key, these functions are activated. Table 2-3 lists several commonly used keys that have special functions in the symbolic language of PDP-8/E computers.

Table 2-3

Special Keyboard Functions

KEY	FUNCTION	USE
SPACE	space	used to combine and delimit symbols or numbers in a symbolic program
RETURN	carriage return	used to terminate line of symbolic program
HERE IS	blank tape	used for leader/trailer (effective only in LOCAL)
RUBOUT	rubout	used for deleting characters, punches all channels on paper tape
CTRL/REPT/P	code 200	used for leader/trailer of binary program paper tapes (keys must be released in reverse order: P, REPT, CTRL)
LINE FEED	line feed	follows carriage return to advance printer one line

PRINTER OPERATION

The printer provides a typed copy of input and output at ten characters per second maximum rate. When the Teletype unit is on line (LINE), the copy is generated by the computer; when the Teletype unit is off line (LOCAL), the copy is automatically generated whenever a key is struck.

PAPER TAPE READER OPERATION

The paper tape reader is used to input into memory data punched on eight-channel perforated paper tape at a maximum rate of ten characters per second. The reader control positions are shown in Figure 2-2 and are described below.

START Activates the reader; reader sprocket wheel is engaged and operative.

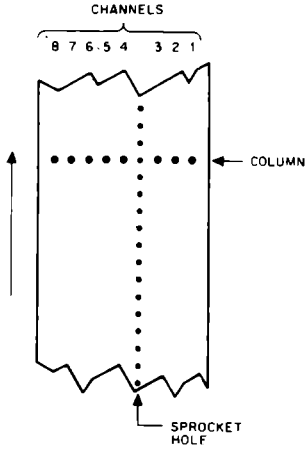
STOP Deactivates the reader; reader sprocket wheel is engaged but not operative.

FREE Deactivates the reader; reader sprocket wheel is disengaged.

PAPER TAPE PUNCH OPERATION

The paper tape punch is used to perforate eight-channel rolled oiled paper tape at a maximum rate of ten characters per second. The punch controls are shown in Figure 2-2 and described below.

REL.	Disengages the tape to allow tape removal or loading.
B. SP.	Backspaces the tape one space for each firm depression of the B. SP. button.
ON	Activates the paper tape punch.
OFF	Deactivates the paper tape punch.



Data is recorded (punched) on paper tape by groups of holes arranged in a definite format along the length of the tape. The tape is divided into channels, which run the length of the tape, and into columns, which extend across the width of the tape, as shown in the adjacent diagram. The paper tape readers and punches used with PDP-8 family computers accept eight-channel paper tape.

Generating a Symbolic Tape

The previously described components may be used to generate a symbolic program paper tape through the following procedure.

When switched to LOCAL, the Teletype unit is independent of the computer and functions like an electric typewriter. Any character struck on the keyboard is printed, and also punched on paper tape if the tape punch is ON. Each character struck on the keyboard is represented in code by one row of holes and spaces according to the ASCII code described in the following section and given in Appendix C.

A section of leader-trailer code several inches long is punched at the beginning of the symbolic tape, by pressing the HERE IS key on the

Teletype keyboard. The symbolic program is then carefully typed, following the conventions used in PDP-8 symbolic programs.

A typing error can be corrected using the B. SP. button of the paper tape punch and the RUBOUT key on the Teletype keyboard. The B. SP. button backspaces the paper tape one column for each depression of the button, and the RUBOUT key perforates all eight channels of a column (this perforation is ignored by the computer). Therefore, errors are removed by backspacing the tape to the error and typing rubouts over the error and all following characters. After typing rubouts, the correct information must be typed beginning where the error occurred.

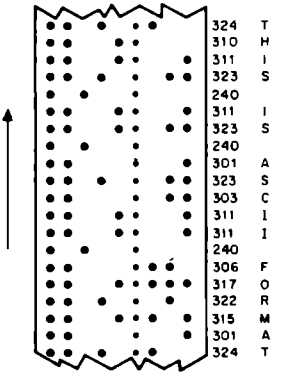
Once the symbolic tape is punched, a form feed is punched, and then more leader-trailer tape is generated by striking the HERE IS key. The tape is removed from the punch unit by tearing against the plastic cover of the punch. The symbolic program thus generated is the input to the assembler described in Chapter 4.

The program may be listed (typed out) by placing the paper tape in the paper tape reader. This is done by releasing the plastic cover of the reader unit and placing the eight-channel tape over the reader head with the smaller sprocket holes over the sprocket wheel, and replacing the cover. If the Teletype control is switched to LOCAL and the reader is switched to START, the tape will advance over the reader head and a printed copy of the program will be typed on the Teletype printer. If the tape punch is also ON, a duplicate of the tape will be generated at the same time.

Paper Tape Formats

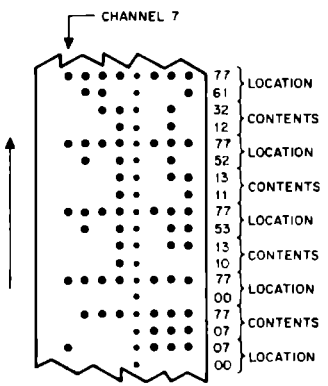
Manual use of the toggle switches on the operator console is a tedious and inefficient means of loading a program. This procedure is necessary in some instances, however, because the PDP-8/E computer must be programmed before any form of input to the memory unit is possible. For example, before any paper tape can be used to input information into the computer, the memory unit must have a stored program which will interpret the paper tape format for the computer. This loader program must be stored in memory with the console switches. A loader program consists of input instructions to accept information from the Teletype paper tape reader and instructions to store the incoming data in the proper memory locations.

Before the loader program can be written to accept information, the format in which the data is represented on the paper tape must be established. There are three basic paper tape formats commonly used in conjunction with PDP-8/E computer. The following paragraphs describe and illustrate these formats.



ASCII FORMAT

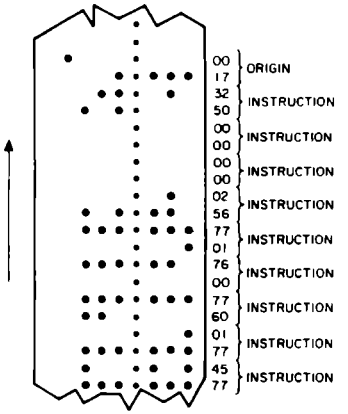
The USA Standard Code for Information Interchange (ASCII) format uses all eight channels* of the paper tape to represent a single character (letter, number, or symbol) as shown in the diagram at left. The complete code is given in Appendix C.



RIM (READ IN MODE) FORMAT

RIM format tape uses adjacent columns to represent 12-bit binary information directly. Channels 1 through 6 are used to represent either address or information to be stored. A channel 7 punch indicates that the adjacent column and the following column are to be interpreted as an address specifying the location in which the information of the following two columns is to be stored. The tape leader and trailer for RIM format tape must be punched in channel 8 only (octal 200).

* Channel 8 is normally designated for parity check. The Teletype units used with the PDP-8/E computer do not generate parity, and Channel 8 is always punched.



BIN (BINARY FORMAT)
BIN format tape is similar to RIM format except that only the first address of consecutive locations is specified. An address is designated by a channel 7 punch and information following an address is stored in sequential locations after the designated address, until another location is specified as an origin. The tape leader/trailer for BIN format tape must be punched in channel 8 (octal 200) only.

Paper Tape Loader Programs

The three previously described paper tape formats are each used for a separate purpose in conjunction with PDP-8/E computer. The ASCII format is used to represent symbolic programs on paper tape, which are then used as input to the assembler. The assembler translates the mnemonic instructions and symbolic addresses into binary instructions and absolute addresses. Once this translation has been performed by the assembler, a binary format tape is generated. ASCII tapes can be quickly recognized by noting that all channels of the tape are used.

The binary format tape is the common means of loading an assembled program into the core memory of a PDP-8/E computer. The BIN (Binary) loader is the program used to load these binary format paper tapes. Program instructions are stored in successive locations beginning with an origin which is signaled by a channel 7 punch on the paper tape. The BIN loader is a lengthy program requiring 83 memory locations. (As an alternative to manually entering the contents of all 83 locations, the RIM (Read In Mode) format is used.) Tapes in binary (BIN) format can be quickly recognized because channel 8 is not used in the middle of the tape, and channel 7 is punched infrequently.

The RIM loader is simpler than the BIN loader because the memory unit is supplied with a location for each incoming instruction. It consists of 17 instructions which must be toggled into memory. The BIN loader is punched in RIM format, and is loaded by the RIM loader; but it is used to load tapes punched in the BIN format, which is the output of the assembly program. Tapes in RIM format are similar in appearance to BIN tapes, but channel 7 is punched every fourth character.

OPERATING PROCEDURES

Many means are available for loading and unloading PDP-8/E information. The means used are dependent upon the form of the information, time limitations, and the peripheral equipment connected to the system. The following procedures are basic to any use of these systems,

and, although they may be used infrequently as the programming and use of the computer become more sophisticated, they are valuable in preparing the initial programs and learning the function of machine input and output transfers.

Manual Data Storage and Modification

Programs and data can be stored or modified manually by means of the facilities on the Programmer's Console. Chief use of manual data storage is made to load the read-in mode (RIM) loader program into the computer core memory. The RIM loader is a program used to automatically load programs into the computer from perforated tape in RIM format. This program and the RIM tape format are described below. Use the following procedure to store data manually in the computer core memory.

Power For Manual Operation

a. Turn the OFF/POWER/PANEL Lock switch clockwise to the POWER position.

Memory Addressing for Manual Operation

a. Set the SWITCH REGISTER switches to correspond to the address bits of the word to be stored.
b. Press the LOAD ADDRESS key.
c. Observe that the address in the switch register is held in the computer as designated by lighted MEMORY ADDRESS indicators corresponding to switches in the 1 (up) position and unlighted indicators corresponding to switches in the 0 (down) position.

Manual Data Input to Addressed Memory Location

a. Set the SWITCH REGISTER switches to correspond to the data or instruction word to be stored at the address just set into the CPMA.
b. Rotate the INDICATOR SELECTOR SWITCH to MD.
c. Lift and release the DEP key.
d. Observe that the data in the SWITCH REGISTER is the same as the data shown on the MD indicators. Data is now stored in the addressed location.
e. Check to see that the MA has been incremented by one so that additional data can be stored at sequential addresses by repeated SWITCH REGISTER settling and deposit key operation.

Checking the Contents of Any Address in Core Memory

a. Perform the Memory Addressing Procedure.
b. Depress the EXAMINE switch.
c. Rotate the INDICATOR SELECTOR switch to the MD position.
d. Observe the data shown on the MD indicators.
e. To observe the next location in core, the contents of the PC and the CPMA are automatically incremented by one. The operator simply depresses the EXAMINE switch and observes the content of the new location.

LOADING DATA UNDER PROGRAM CONTROL

Information can be stored or modified automatically in the computer only by using programs previously stored in core memory. For example,

having the RIM loader stored in core memory allows RIM format tape to be loaded as follows:

INITIALIZING THE SYSTEM

a. Rotate the OFF/POWER/PANEL LOCK switch clockwise to the POWER position.
b. Set the Teletype LINE/OFF/LOCAL switch to the LINE position.
c. Load the tape in the Teletype reader by settling the START/STOP/FREE switch to the FREE position, releasing the cover guard by means of the latch at the right, loading the tape so that the sprocket wheel teeth engage the feed holes in the tape, closing the cover guard, moving the tape either forward or backward until the punched leader section is over the read station, and setting the switch to the STOP position. Tape is loaded in the back of the reader so that it moves toward the front as it is read. Proper positioning of the tape in reader results in three bit positions being sensed to the left of the sprocket wheel and five bit positions being sensed to the right of the sprocket wheel. The directional arrow printed on the tape should point toward the operator.
d. Load the starting address of the RIM loader program (not the address of the program to be loaded) into the PC by means of the Switch Register and the LOAD ADDRESS switches.
e. In sequence, press the computer keys CLEAR and CONTINUE and set the 3-position Teletype reader switch to the START position. The tape is then read automatically.
f. Stop the computer program by means of the Halt switch when the reader reaches the trailer section of tape.

PROGRAM LOADING OPERATION

Automatic storing of the binary loader (BIN) program is performed by means of the RIM loader program as described below. With the BIN loader stored in core memory, program tapes assembled in the program assembly language (PAL III) binary format can be stored as described in the previous procedure, except that the starting address of the BIN loader (usually 7777) is used in step d. When the BIN program is loaded the computer stops; at this point the AC should contain all zeros if the program is stored properly. If the computer stops with a number other than zero in the AC, a checksum error has been detected. When the program has been stored, it can be initiated by loading the program starting address (usually designated on the leader of the tape) into the PC by means of the Switch Register and LOAD ADDRESS switches, then pressing and releasing the CLEAR key and then pressing the CONTinue key.

The steps involved in the process of loading programs and bringing the system up to the point where the user can communicate with the processor is illustrated in Figure 2-4. The loading flow diagram for each type of program to be loaded is referenced, and each flow diagram is accompanied with a corresponding procedure.

This loading procedure is greatly simplified when the user employs a mass storage device such as the Disk Monitor System. Using the Monitor, stored programs are called in from Disk files. This is illustrated in Figure 2-5.

Loaders

When a PDP-8/E computer is first received, it should be assumed that no useful information is in memory. The machine is not capable of performing any arithmetic operations or receiving data.

All tapes in the Program Library are written in binary format. The user, therefore, must load the machine so that it is capable of accepting binary tapes. Initially, sixteen RIM instructions are manually toggled into memory. The binary loader tape is then used to eliminate the necessity of toggling in 86 additional instructions.

RIM allows the binary loader tape to be read into memory and the binary loader tape allows the use of any tape in the Program Library such as symbolic editor.

READ-IN-MODE (RIM) LOADER

The RIM Loader is the very first program loaded into the computer, and it is loaded by the programmer using the console switches. The RIM Loader instructs the computer to receive and store, in core, data punched on paper tape in RIM coded format. (RIM Loader is used to load the BIN Loader described below.)

There are two RIM loader programs: one is used when the input is to be from the low-speed paper tape reader, and the other is used when input is to be from the high-speed paper tape reader. The locations and corresponding instructions for both loaders are listed in Table 2-4.

The procedure for loading (toggling) the RIM Loader into core is illustrated in Figure 2-6.

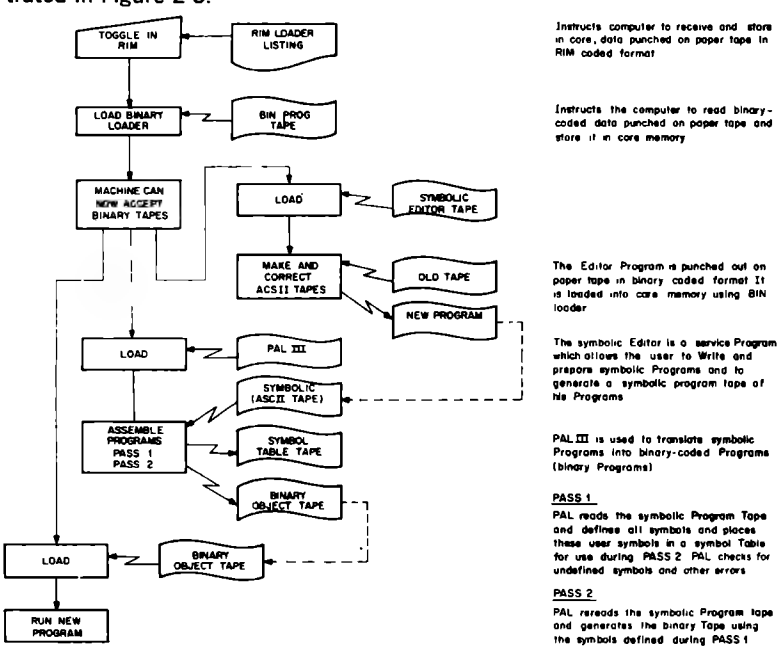

TOGGLE IN RIM
RIM LOADER LISTING
Instructs computer to receive and store in core, data punched on paper tape in RIM coded format
LOAD BINARY LOADER
BIN PROG TAPE
Instructs the computer to read binary-coded data punched on paper tape and store it in core memory
MACHINE CAN NOW ACCEPT BINARY TAPES
LOAD
SYMBOLIC EDITOR TAPE
MAKE AND CORRECT ACSII TAPES
OLD TAPE
NEW PROGRAM
The Editor Program is punched out on paper tape in binary coded format It is loaded into core memory using BIN loader
LOAD
PAL III
SYMBOLIC (ASCII TAPE)
The symbolic Editor is a service Program which allows the user to Write and prepare symbolic Programs and to generate a symbolic program tape of his Programs
ASSEMBLE PROGRAMS PASS 1 PASS 2
SYMBOL TABLE TAPE
BINARY OBJECT TAPE
PAL III is used to translate symbolic Programs into binary-coded Programs (binary Programs)
PASS 1
PAL reads the symbolic Program Tape and defines all symbols and places these user symbols in a symbol Table for use during PASS 2 PAL checks for undefined symbols and other errors
PASS 2
PAL rereads the symbolic Program tape and generates the binary Tape using the symbols defined during PASS 1
LOAD
BINARY OBJECT TAPE
RUN NEW PROGRAM

Figure 2-4 Loading Data Under Program Control

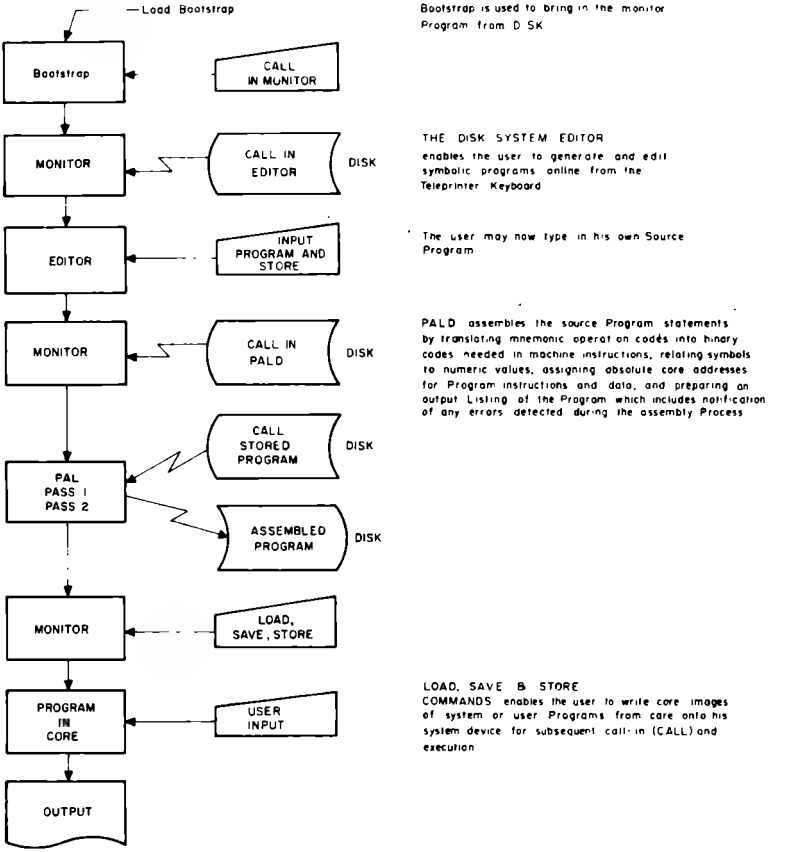

—Load Bootstrap
Bootstrap is used to bring in the monitor Program from D SK
Bootstrap
CALL IN MONITOR
MONITOR
CALL IN EDITOR
DISK
THE DISK SYSTEM EDITOR enables the user to generate and edit symbolic programs online from the Teleprinter Keyboard
EDITOR
INPUT PROGRAM AND STORE
The user may now type in his own Source Program
MONITOR
CALL IN PALD
DISK
PALD assembles the source Program statements by translating mnemonic operat on codes into binary codes needed in machine instructions, relating symbols to numeric values, assigning absolute core addresses for Program instructions and data, and preparing an output Listing of the Program which includes notification of any errors detected during the assembly Process
CALL STORED PROGRAM
DISK
PAL PASS 1 PASS 2
ASSEMBLED PROGRAM
DISK
MONITOR
LOAD, SAVE, STORE
PROGRAM IN CORE
USER INPUT
LOAD, SAVE & STORE COMMANDS enables the user to write core images of system or user Programs from core onto his system device for subsequent call-in (CALL) and execution
OUTPUT

Figure 2-5 Loading Programs with Mass Storage Devices

Table 2-4
RIM Loader Programs

LOCATION	INSTRUCTION	
	Low-Speed Reader	High-Speed Reader
7756	6032	6014
7757	6031	6011
7760	5357	5357
7761	6036	6016
7762	7106	7106
7763	7006	7006
7764	7510	7510
7765	5357	5374
7766	7006	7006

LOCATION	INSTRUCTION	
	Low-Speed Reader	High-Speed Reader
7767	6031	6011
7770	5367	5367
7771	6034	6016
7772	7420	7420
7773	3776	3776
7774	3376	3376
7775	5356	5357
7776	0000	0000

After RIM has been loaded, it is good programming practice to verify that all instructions were stored properly. This can be done by performing the steps illustrated in Figure 2-7, which also shows how to correct an incorrectly stored instruction.
When loaded, the RIM Loader occupies absolute locations 7765 through 7776.

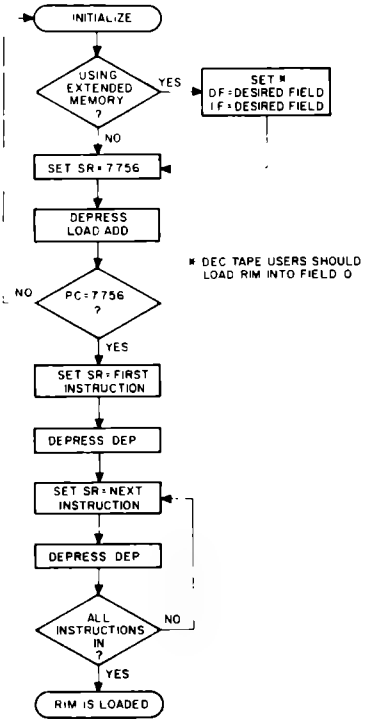


Figure 2-6 Loading the RIM Loader

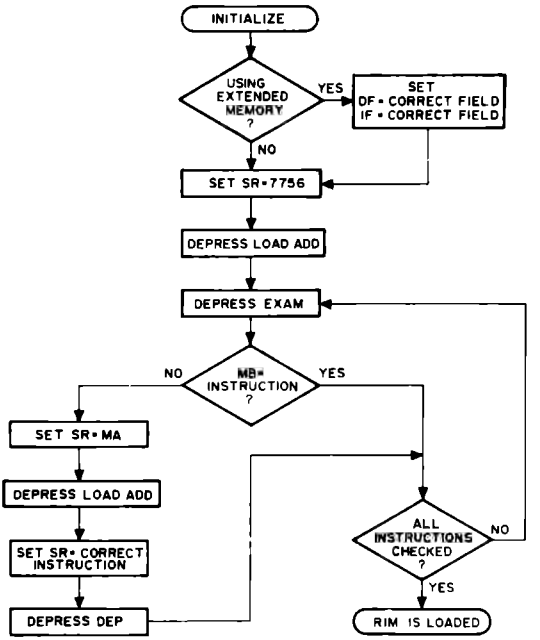


Figure 2-7 Checking the RIM Loader

BINARY (BIN) LOADER

The BIN Loader is a short utility program which, when in core, instructs the computer to read binary-coded data punched on paper tape and store it in core memory. BIN is used primarily to load the programs furnished in the software package (excluding the loaders and certain subroutines) and the programmer's binary tapes.

BIN is furnished to the programmer on punched paper tape in RIM-coded format. Therefore, RIM must be in core before BIN can be loaded. Figure 2-8 illustrates the steps necessary to properly load BIN. When loading, the input device (low- or high-speed reader) must correspond to the version of RIM loaded in the machine.

When stored in core. BIN resides on the last page of core, occupying absolute locations 7625 through 7752 and 7777.

BIN was purposely placed on the last page of core so that it would always be available for use—the programs in DEC's software package do not use the last page of core (excluding the Disk Monitor). The programmer must be aware that if he writes a program which uses the last page of core, BIN will be wiped out when that program runs on the computer. When this happens, the programmer must load RIM and then BIN before he can load another binary tape.

Figure 2-9 illustrates the procedure for loading binary tapes into core.

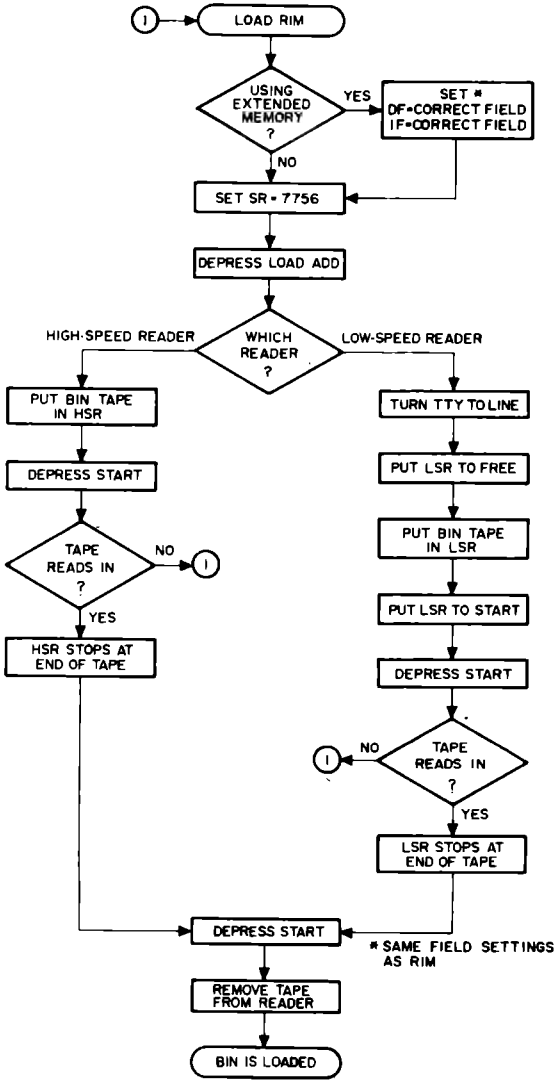


Figure 2-8 Loading the BIN Loader

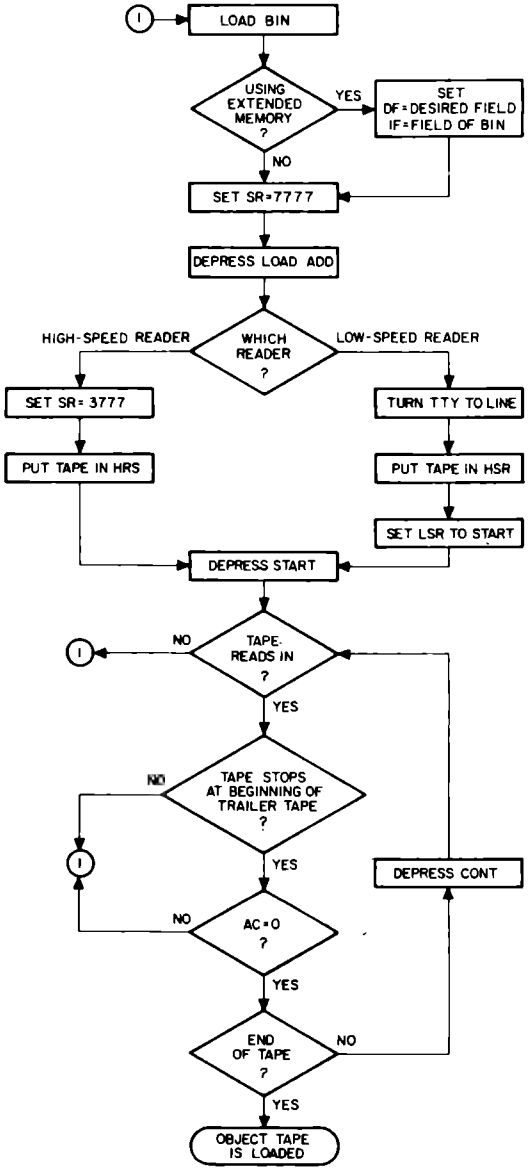


Figure 2-9 Loading A Binary Tape Using BIN

Symbolic Editor

The Symbolic Editor is a service program which allows the programmer to write and prepare symbolic programs and to generate a symbolic program tape of his programs. Editor is very flexible in that the programmer can type his symbolic program online from the Teletype keyboard, thus storing it directly into core memory. Then, using certain Editor commands, the programmer can have his program listed (printed) on the teleprinter for visual inspection.

Editor also allows the programmer to add, correct, or delete any portion of his symbolic program. When the programmer is satisfied that his program is correct and ready to be assembled or compiled, Editor can be commanded to generate a symbolic program tape of the stored program.

The Symbolic Editor program is issued on punched paper tape in binary-coded format. Therefore, it is loaded into core memory using the BIN Loader. When in core, Editor is activated for use by setting the switch register (SR) to 0200 (the starting address) and depressing the LOAD ADD (load address) and then START switches. Editor responds with a carriage return/line feed sequence on the Teletype.

Initially, Editor is in command mode, that is, it is ready to accept commands from the programmer; anything typed by the programmer is interpreted as a command to Editor. Editor accepts only legal commands, and if the programmer types something else, Editor ignores the command and types a question mark (?).

When not in command mode, Editor is in text mode, that is, all characters typed from the keyboard or tapes read in on the tape reader are interpreted as text to be put into the text buffer in the manner specified by a preceding Editor command. Figure 2-10 illustrates how the programmer can transfer Editor from one mode to another.

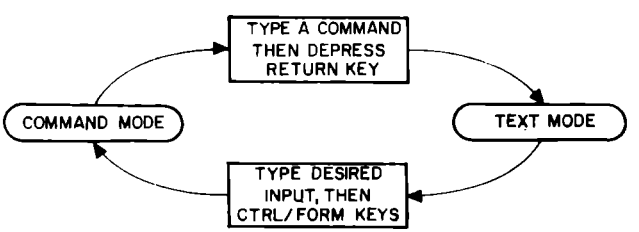


Figure 2-10 Transition Between Editor Modes

Seven of Editor's basic commands are briefly described below.

COMMAND	MEANING
A	Append incoming text from the keyboard into the text buffer immediately following the text currently stored in the buffer.
R	Read incoming text from the tape reader and append it to the text currently stored in the buffer.
L	List entire text buffer; the programmer can specify one line or a group of lines.
C	Change a line; the programmer precedes the command with the decimal line number or line numbers of the lines to be changed.
I	Insert into text buffer; the programmer specifies the decimal line number in his program where the inserted text is to begin.
D	Delete from text buffer; the programmer specifies the line or group of lines to be deleted.
P	Punch text buffer; the programmer can specify one line, a group of lines, or the entire text buffer.

All commands are executed when the RETURN key is depressed except the P command. To execute the P command, press the RETURN key on the Teletype, turn on the punch, and press the CONT (continue) switch on the computer console.

The above commands are only the seven basic commands. A summary of all commands is provided in Table 2-7 at the end of this section.

WRITING A PROGRAM

Now that you have some idea of what you can do with Editor and what Editor can do for you, we will write and edit a short program, explaining each step in the comments to the right of the printout.

The example program finds the larger of two numbers and halts with the number displayed in the accumulator (AC). The program is written in PAL III, to be assembled using the PAL III Assembler described later in this chapter.

The programmer loads Editor using the BIN loader (see Figure 2-8). Editor is then activated by loading the starting address (0200(octal)) and depressing the LOAD ADD and START switches. After Editor responds with a carriage return-line feed, the programmer types A and RETURN key, Editor is now in text mode, that is, subsequent characters typed are appended to the text buffer. The programmer now types the symbolic program. (Block indenting is facilitated using the CTRL/TAB key, which Editor has programmed to indent in ten-character increments.)

```
A
                *200
                CLA             /CLEAR AC
                TAD NUMB        /GET B
                CMA
                CMA             /1'S COMP B
                IAC             /-B
                TAD NUMA        /ADD -B + A
                SMA             /IF -B LARGER
                JMP .+4         /JUMP 4 LOCATIONS
                CLA             /CLEAR AC
                TAD NUMB        /GET B
                HLT             /B IS LARGER
                CLA             /CLEAR AC
                TAD NUMA        /GET A
                HLT             /A IS LARGER
NUMB,           0000
NUMA,           0000
                $
```

Visual inspection reveals that we have errors in lines 4, 16, and 17. (Editor maintains a line number count in decimal, with the first line typed being 1 and our last line being 18.) Line 4 can be removed using the D (Delete) command, and lines 16 and 17 can be corrected using the C (Change) command. However, Editor is presently in text mode, and in order to issue another command Editor must be transferred to command mode. This is done when the programmer types CTRL/FORM (depress and hold down the CTRL key while typing the FORM key).

CTRL/FORM (nonprinting)	The programmer types CTRL/FORM; Editor responds with CR/LF and rings the teleprinter bell, indicating that it is in command mode.
4D	The programmer types 4D and the RETURN key; Editor responds with a CR/LF and the line is deleted.
15, 16C	The programmer types 15, 16C and the RETURN key, informing Editor that lines 15 and 16 (formerly 16 and 17) are to be changed. Editor responds with a CR/LF, transfers to text mode, and waits for the programmer to change the lines.
NUMA, 1111 NUMB, 0011	The programmer types NUMA, 1111 and NUMB, 0011.

The symbolic program should now be correct. However, it is good programming practice to check the program after editing; this can be done using the L (List) command, but since only original lines 4, 16, and 17 were changed it is not necessary to have the whole program listed. The programmer can command Editor to list lines 4 through 17.

CTRL/FORM (nonprinting)	The programmer types CTRL/FORM to return Editor to command mode; Editor responds with CR/LF and rings the bell, and waits for the next command.
4, 17L	The programmer types 4, 17L and the RETURN key; Editor types lines 4 through 17.

```
        CMA         /1'S COMP B
        IAC         /-B
        TAD NUMA    /ADD -B + A
        SMA         /IF -B LARGER
        JMP .+4     /JUMP 4 LOCATIONS
        CLA         /CLEAR AC
        TAD NUMB    /GET B
        HLT         /B IS LARGER
        CLA         /CLEAR AC
        TAD NUMA    /GET A
        HLT         /A IS LARGER
NUMA,   1111
NUMB,   0011
        $
```

The changes were accepted properly. The symbolic program is correct and ready to be punched on paper tape.

GENERATING A PROGRAM TAPE

Before issuing the P (Punch) command, Editor must be in command mode. Figure 2-11 illustrates the procedures required to generate a symbolic program tape using Editor.

CTRL/FORM (nonprinting)	The programmer types CTRL/FORM; Editor responds with a question mark, indicating that Editor was already in command mode.
?	
P	The programmer commands Editor to punch the entire text buffer by typing P and the RETURN key.

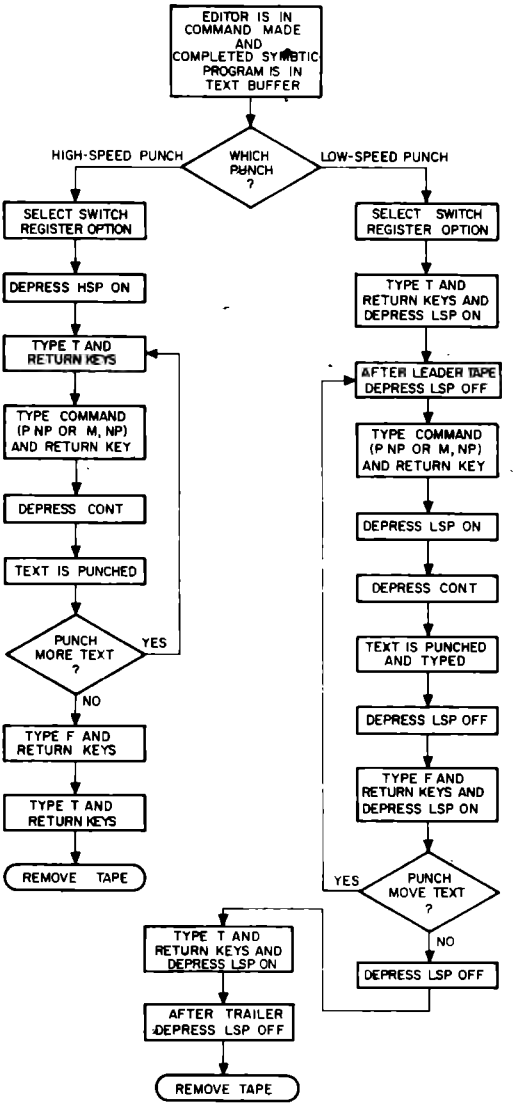

EDITOR IS IN COMMAND MADE AND COMPLETED SYMBOLIC PROGRAM IS IN TEXT BUFFER
WHICH PUNCH ?
HIGH-SPEED PUNCH
LOW-SPEED PUNCH
SELECT SWITCH REGISTER OPTION
DEPRESS HSP ON
TYPE T AND RETURN KEYS
TYPE COMMAND (P NP OR M, NP) AND RETURN KEY
DEPRESS CONT
TEXT IS PUNCHED
PUNCH MORE TEXT ?
YES
NO
TYPE F AND RETURN KEYS
TYPE T AND RETURN KEYS
REMOVE TAPE
SELECT SWITCH REGISTER OPTION
TYPE T AND RETURN KEYS AND DEPRESS LSP ON
AFTER LEADER TAPE DEPRESS LSP OFF
TYPE COMMAND (P NP OR M, NP) AND RETURN KEY
DEPRESS LSP ON
DEPRESS CONT
TEXT IS PUNCHED AND TYPED
DEPRESS LSP OFF
TYPE F AND RETURN KEYS AND DEPRESS LSP ON
PUNCH MOVE TEXT ?
YES
NO
DEPRESS LSP OFF
TYPE T AND RETURN KEYS AND DEPRESS LSP ON
AFTER TRAILER DEPRESS LSP OFF
REMOVE TAPE

Figure 2-11 Generating a Symbolic Tape Using Editor

When Editor recognizes a P command it waits for the programmer to specify the low- or high-speed punch. If the programmer wants the pro-

gram punched and typed, he sets SR bit 10 to 0 and the program will be punched on the low-speed punch and simultaneously typed on the teleprinter. If the programmer want only a program tape and if he has a high-speed punch available, he sets SR bit 10 to 1 and the program will be punched on the high-speed punch. For the purposes of this discussion, a printed program listing is desired, so the low-speed punch is specified. The programmer turns on the low-speed punch and depresses the CONT switch on the computer console, and Editor begins punching and typing the contents of the entire text buffer.

An image of the stored symbolic program has been punched and typed by Editor.

If the programmer stops the computer, e.g., purposely or accidentally turning the computer off, he may restart Editor at location 0200 or 0177 without disturbing the text in the buffer. Editor can also be restarted at location 0176; however, all text currently in the buffer is wiped out. Therefore, the programmer can restart at location 0176 to re-initialize for a new program.

SEARCH FEATURE

A very convenient feature available with Editor is the search feature which allows the programmer to search a line of text for a specified character. When the programmer types a line number followed by S, Editor waits for the user to type in the character for which it is to search. The search character is not echoed (printed on the teleprinter). When Editor locates and types the search character typing stops, and Editor waits for the programmer to either type new text and terminate the line with a RETURN key or to use one of the following special keys.

1.	←	to delete the entire line to the left,
2.	RETURN	to delete the entire line to the right,
3.	RUBOUT	to delete from right to left one character for each RUBOUT typed (a / is echoed for each RUBOUT typed).
4.	LINE FEED	to insert a carriage return/line feed (CR/LF) thus dividing the line into two,
5.	CTRL/FORM	to search for the next occurrence of the search character, and/or
6.	CTRL/BELL	to change the search character to the next character typed by the programmer.

INPUT/OUTPUT CONTROL

Switch register options are used with input and output commands to control the reading and punching of paper tape. The options available to the programmer are shown in Table 2-5. These options are used in conjunction with the "Select Switch Register Option" operation in Figure 2-11.

Table 2-5

Input/Output Control

SR BIT	POSITION	FUNCTION
0	0	Input text as is
	1	Convert all occurrences of 2 or more spaces to a tab
1	0	Output each tab as 8 spaces
	1	Tab is punched as tab/rubout
2	0	Output as specified
	1	Suppress output*
10	0	Low-speed punch and Teleprinter
	1	High-speed punch
11	0	Low-speed reader
	1	High-speed reader

* Bit 2 allows the user to interrupt any output command and return immediately to command mode; when desired, merely set bit 2 to 1.

ERROR DETECTION

Editor checks all commands for nonexistent information and incorrect formatting. When an error is detected, Editor types a question mark (?), and ignores the command. However, if an argument is provided for a command that doesn't require one, the argument is ignored and the command is executed properly.

Editor does not recognize extraneous and illegal control characters; therefore, a tape containing these characters can be cleared up or corrected by merely reading the tape into Editor and punching out a new tape.

SUMMARY OF SPECIAL KEYS AND COMMANDS

Using special keyboard keys and commands, the programmer controls Editor's operation. Certain keys have special meaning to Editor, of which some can be used in either command or text mode. The mode of operation determines the function of each key. The special keys and their functions are shown in Table 2-6.

Table 2-6

Special Keys

KEY	COMMAND MODE	TEXT MODE
RETURN	Execute preceding command	Enter line in text buffer
←	Cancel preceding command (Editor responds with a ? followed by a carriage return and line feed)	Cancel line to the left margin

Table 2-6 (Cont.)

Special Keys

KEY	COMMAND MODE	TEXT MODE
RUBOUT	same as ←	Delete to the left one character for each depression; a \ (backslash) is echoed (not used in Read (R) command)
CTRL/FORM	Respond with question mark and remain in command mode	Return to command mode and ring teleprinter bell
. (period)	Value equal to decimal value of current line (may be used alone or with + or — and a number, e.g., .+8)	Legal text character
/	Value equal to number of last line in buffer; used as an argument	Legal text character
LINE FEED	List next line	Used in Search (S) command to insert CR/LF into line
ALTMODE	List next line	
>	List next line	
<	List previous line	
=	Used with . or / to obtain their value	
	Same as = (gives value of legitimate argument)	
CTRL/TAB		Produces a tab which on output is interpreted as 10 spaces or a tab/rubout, depending on SR option

Editor commands are given when in command mode. There are three basic types of commands: Input, Editing, and Output. Table 2-7 contains a summary of Editor commands and their function.

Table 2-7

Summary of Commands

TYPE	COMMAND	FUNCTION
Input	A	Append incoming text from keyboard into text buffer
	R	Append incoming text from tape reader into text buffer
Editing	L	List entire text buffer
	nL	List line n
	m,nL	List lines m through n inclusively
	nC	Change line n
	m,nC	Change lines m through n inclusively
	I	Insert before first line
	nI	Insert before line n
	K	Delete entire text buffer
	nD	Delete line n
	m,nD	Delete lines m through n inclusively
	m,n$jM	Move lines m through n to before line j
	G	Print next tagged line (if none, Editor types ?)
	nG	Print next tagged line after line n (if none, ?)
	S	Search buffer for character specified after RETURN key and allow modification (search character is not echoed on printer)
	nS	Search line n, as above
	m,nS	Search lines m through n inclusively, as above
Output	P	Punch entire text buffer
	nP	Punch line n
	m,nP	Punch lines m through n inclusively
	T	Punch about 6 inches of leader/trailer tape
	F	Punch a FORM FEED onto tape
	N	Do P, F, K, and R commands

m and n are decimal numbers, and m is smaller than n; j is a decimal number.

The P and N commands halt the Editor to allow the programmer to select I/O control; press CONT to execute these commands.

Commands are executed when the RETURN key is depressed, excluding the P and N commands.

PAL III SYMBOLIC ASSEMBLER

The PAL III Symbolic Assembler (PAL stands for Program Assembly Language) is an indispensable service program used to translate symbolic programs, which are written in the PAL III language, into binary-coded programs (binary programs).

PAL III is a two-pass assembler with an optional third pass, i.e., the symbolic program tape must be passed through the assembler two times to produce the binary-coded tape (binary tape), and the optional third pass produces a complete octal/symbolic program listing which can be

typed and/or punched if desired. A brief explanation of the three passes is given below.

Pass 1. The assembler reads the symbolic program tape and defines all symbols used and places the user symbols in a symbol table for use during Pass 2. The assembler checks for undefined symbols and certain other errors and types an error message on the teleprinter when an error is detected.

Pass 2. The assembler rereads the symbolic program tape and generates the binary tape using the symbols defined during Pass 1. When the low-speed punch is used, meaningless characters will be typed on the teleprinter, and these should be ignored by the programmer. The assembler checks illegal referencing during this pass and types an error message on the teleprinter when any is detected.

Pass 3. The assembler reads the symbolic program tape and types and/or punches the octal/symbolic program assembly listing. This listing thoroughly documents the assembled program and is useful when debugging and modifying the program.

The meaningless characters, error messages, and octal/symbolic program listing will be shown later in this section.

PAL III accepts symbolic program tapes from either the low-speed or high-speed reader and produces the binary tapes on either the low-speed or high-speed punch.

During assembly, the programmer communicates with PAL III via the switches on the computer console. Switch options are used to specify which pass the assembler is to perform and which reader and punch the assembler should accept input from and punch out on.

ASSEMBLING A SYMBOLIC PROGRAM

Earlier in this chapter, the programmer wrote a PAL III symbolic program and generated the symbolic program tape using Editor. That symbolic program can now be assembled to produce a binary program using PAL III. A listing of the symbolic program follows.

```
        *200
        CLA             /CLEAR AC
        TAD NUMB        /GET B
        CMA             /1'S COMP B
        IAC             /-B
        TAD NUMA        /ADD -B + A
        SMA             /IF -B LARGER
        JMP .+4         /JUMP 4 LOCATIONS
        CLA             /CLEAR AC
        TAD NUMB        /GET B
        HLT             /B IS LARGER
        CLA             /CLEAR AC
        TAD NUMA        /GET A
        HLT             /A IS LARGER
NUMA,   1111
NUMB,   0011
        $
```

First, PAL III must be loaded into core memory, and since PAL III is on punched paper tape in binary-coded format, it is loaded into core memory using the BIN Loader (see Figure 2-8 for loading procedures).

With PAL III in core, we are ready to assemble the symbolic program. Figures 2-12 and 2-13 illustrate the procedures for assembling with PAL III using the low-speed reader/punch and high-speed reader/punch, respectively. In these flowcharts, the switch register options are set for the appropriate reader/punch.

The low-speed reader and punch (LSR and LSP) are used in the following assembly (see Figure 2-13).

Initializing and Starting	Load PAL III into core memory using BIN. Set SR = 0200 and depress LOAD ADD. Turn TTY to LINE and put symbolic program tape in LSR.
Entering Pass 1	Set SR = 2200 and set LSR to START. Turn LSP ON and depress START.
NUMA 0215 NUMB 0216	Error messages would be typed now. Symbol table concludes Pass 1. Examine Symbol table. Make sure no "UA" diagnostics appear.
Entering Pass 2 BB: 8 8 = * < : <)	Put symbolic program tape in LSR. Set SR = 4200 and set LSR to START. Depress LSP to ON and depress CONT. Disregard meaningless characters while object tape is being punched. Error message would be typed now.
Entering Pass 3	Put symbolic program tape in LSR. Set SR = 6200 and set LSR to START. Depress LSP to ON and depress CONT. The octal/symbolic program listing is being typed and punched.

```
                        .*200
0200    7200            CLA         /CLEAR AC
0201    1216            TAD NUMB    /GET B
0202    7040            CMA         /1'S COMP B
0203    7001            IAC         /-B
0204    1215            TAD NUMA    /ADD -B + A
0205    7500            SMA         /IF -B LARGER
0206    5212            JMP .+4     /JUMP 4 LOCATIONS
0207    7200            CLA         /CLEAR AC
0210    1216            TAD NUMB    /GET B
0211    7402            HLT         /B IS LARGER
0212    7200            CLA         /CLEAR AC
0213    1215            TAD NUMA    /GET A
0214    7402            HLT         /A IS LARGER
0215    1111    NUMA,   1111
0216    0011    NUMB,   0011
NUMA    0215
NUMB    0216
```

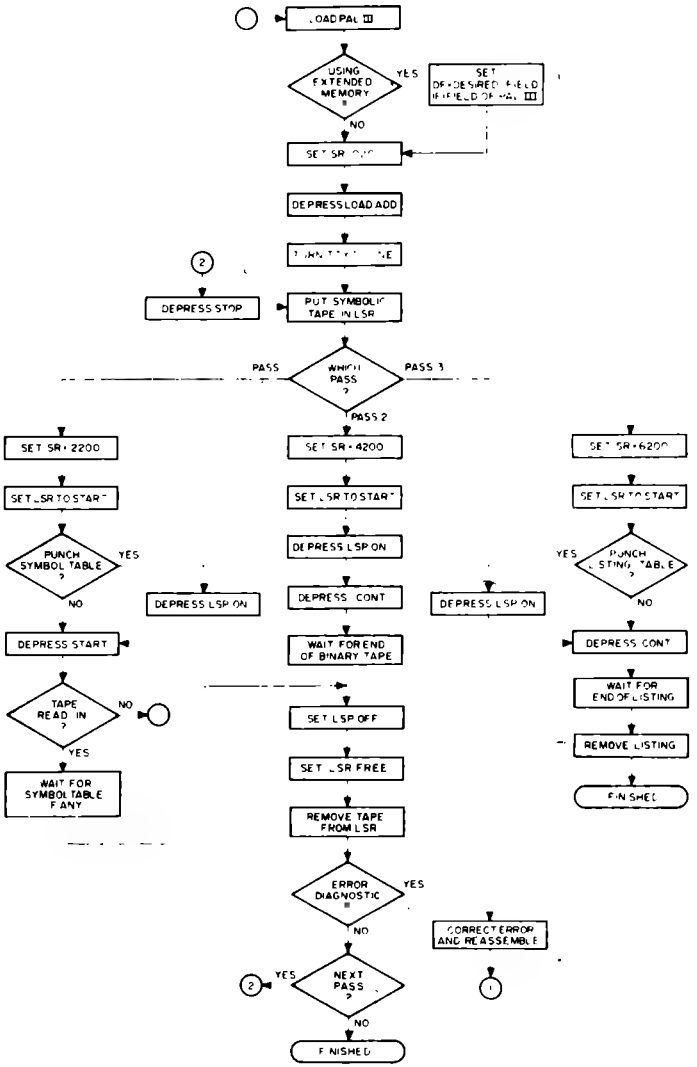


Figure 2-12 Assembling with PAL III Using Low-Speed Reader/Punch

The tape produced during Pass 2 is the binary tape, which is loaded into core memory using the BIN Loader. The symbol table tape produced during Pass 1, the binary tape produced during Pass 2, and the octal/symbolic program listing produced during Pass 3 are used when debugging the program.

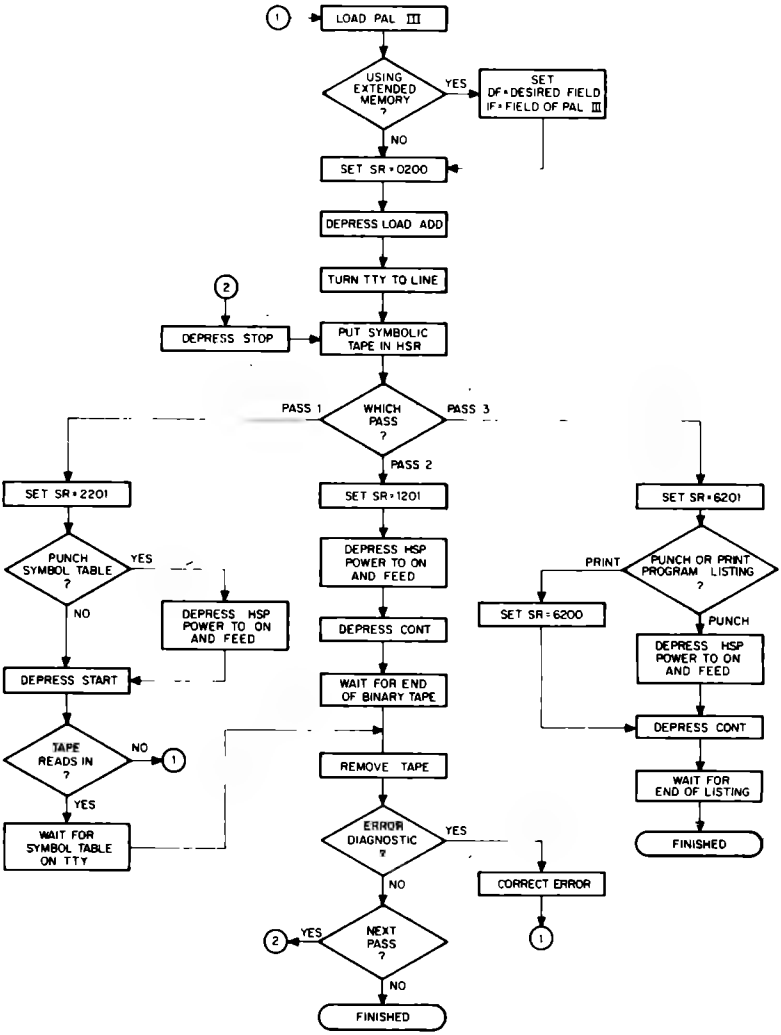


Figure 2-13 Assembling with PAL III Using High-Speed Reader/Punch

Off-Line Teletype Operation

The Teletype can be used separately from the computer for typing, punching tape, or duplicating tapes. To use the Teletye in this manner:

a. Ensure that the computer PANEL LOCK switch is positioned to the PANEL LOCK position.

b. Set the Teletype LINE/OFF/LOCAL switch to the LOCAL position.
c. If the punch is to be used, load it by raising the cover, manually feeding the tape from the top of the roll into the guide at the back of the punch, advancing the tape through the punch by manually turning the friction wheel, and then closing the cover. Energize the punch by pressing the ON pushbutton, and produce about two feet of leader. The leader-trailer can be code 200 or 377. To produce the code 200 leader, simultaneously press and hold the CTRL and SHIFT keys with the left hand, press and hold the REPT key, and press the @ (P) key. When the required amount of leader has been punched, release the @ key, and then all keys. To produce the 377 code, simultaneously press and hold both the REPT and RUB OUT keys until a sufficient amount of leader has been punched.

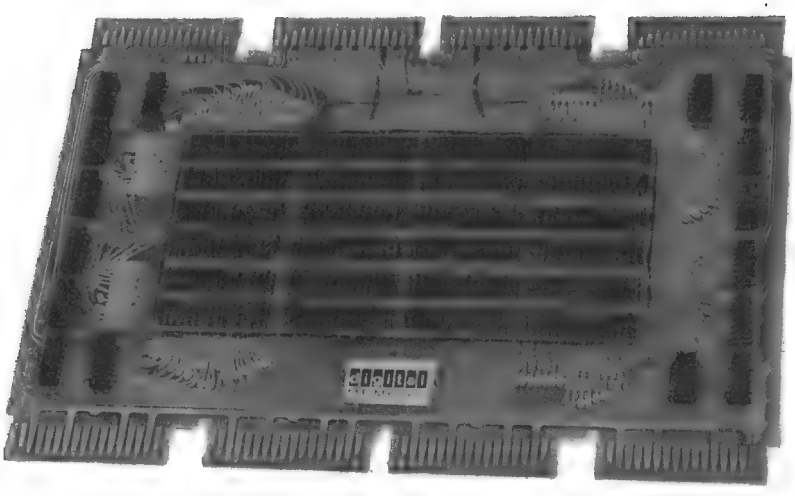
If an incorrect key is struck while punching a tape, the tape can be corrected as follows: If the error is noted after typing and punching any number (n) of characters, press the punch B.SP. (backspace) pushbutton n + 1 times and strike the keyboard RUB OUT key n + 1 times. Then continue typing and punching with the character which was in error.

To duplicate and obtain a listing of an existing tape: Perform the procedure under steps a through c above. Then load the tape to be duplicated as described in step b of the procedure listed under Loading Data Under Program Control. Initiate tape duplication by setting the reader START/STOP/FREE switch to the START position. The punch and teleprinter stops when the tape being duplicated is completely read. Corrections to insert or delete information on a perforated tape can be made by duplicating the correct portion of the tape and manually punching additional information, or inhibiting punching of information to be deleted. This is accomplished by duplicating the tape and carefully observing the information being typed as the tape is read. In this manner, the reader START/STOP/FREE switch can be set to the STOP position just before the point of the correction is reached. Information to be inserted can then be punched manually by means of the keyboard. Information can be deleted by pressing the punch OFF pushbutton and operating the reader until the portion of the tape to be deleted has been typed. It may be necessary to backspace and rub out one or two characters on the new tape if the reader is not stopped precisely on time. The number of characters to be rubbed out can be determined exactly by the typed copy. Be sure to count spaces when counting typed characters. Continue duplicating the tape in the normal manner after making the corrections.

New, duplicated, or corrected perforated tapes should be verified by reading them off-line and carefully proofreading the typed copy.

Program Control

When the program is stopped at the end of an instruction with the single step key, then the load address, examine, and deposit keys may be used without changing the AC. The program may then be resumed by resetting the PC to the address wanted, and then operating the CONTinue key.



4K memory

CHAPTER 3

MEMORY AND PROCESSOR INSTRUCTIONS

GENERAL

An instruction is a coded program step that tells the computer what to do for a single operation in a program. In the PDP-8 family of computers, there are two major types of instruction words: memory reference and augmented. Memory reference instructions store or retrieve data from core memory, while augmented instructions do not. A third type—a housekeeping instruction is defined later. All instructions are determined by bits 0 through 2 to specify the operation (op) code. Operation codes of 0(octal) through 4(octal) specify memory reference instructions, and code of 6(octal) and 7(octal) specify augmented instructions. Memory reference instruction times are 1.2 μs for instruction fetches and non-auto-indexed defer cycles, and 1.4 μs for all other cycles. IOT (Input/Output Transfer) instructions are 1.2 μs for options which communicate directly with the OMNIBUS; and 2.6, 3.6, or 4.6 μs for options which communicate via the KA8-E Positive I/O Bus Interface. The latter times are + or −5%, all other times are + or −1%.

The instruction repertoire is shown in Figure 3-1. This illustrates the eight basic instructions and the division into the three categories—the Memory Reference Instructions, Augmented Instructions, and the Housekeeping Instruction.

The Memory Reference Instructions are concerned with at least two memory addresses, the address of the initial memory location and the address of the data. Figure 3-2 illustrates an example of the address flow of a Memory Reference Instruction. In order to illustrate the basic flow, a simplified diagram is shown leaving out some of the more complex branching such as JUMP and AUTO INDEX instructions. Two cycles are required (at a minimum) to complete an instruction. The indirect addressing capability requires an additional cycle (DEFER) which must go back to memory for another address. On the next cycle (EXECUTE), the data is brought into the Central Processor and the instruction is completed.

The Augmented Instruction simplified flow diagram is illustrated in Figure 3-3. Notice that the instruction is completely performed in the FETCH cycle. Once memory is addressed and the instruction brought from memory to the central processor, memory is not referenced (i.e., no additional instruction is brought from memory). Instead, the instruction is completely carried out once the initial decoding has been accomplished. For Augmented instructions, decoding will bring either operate instructions or IOT instructions. The last 9 bits combined with the logic bring about the required results.

The Housekeeping Instruction, JMP is used only to force any desired address into the Program Counter. (See Program Control in chapter 4)

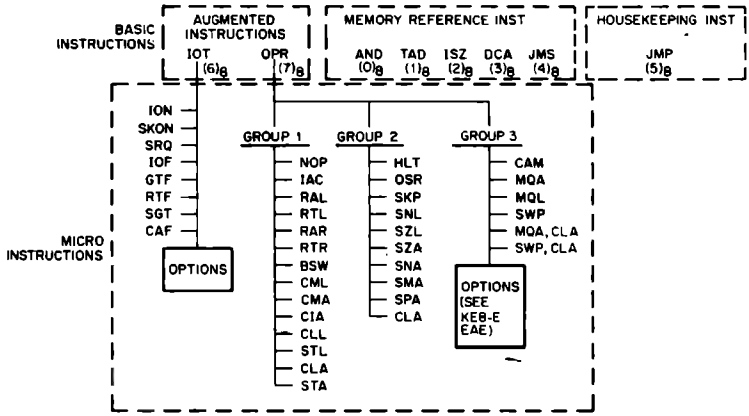


Figure 3-1 PDP-8/E Instruction Repertoire

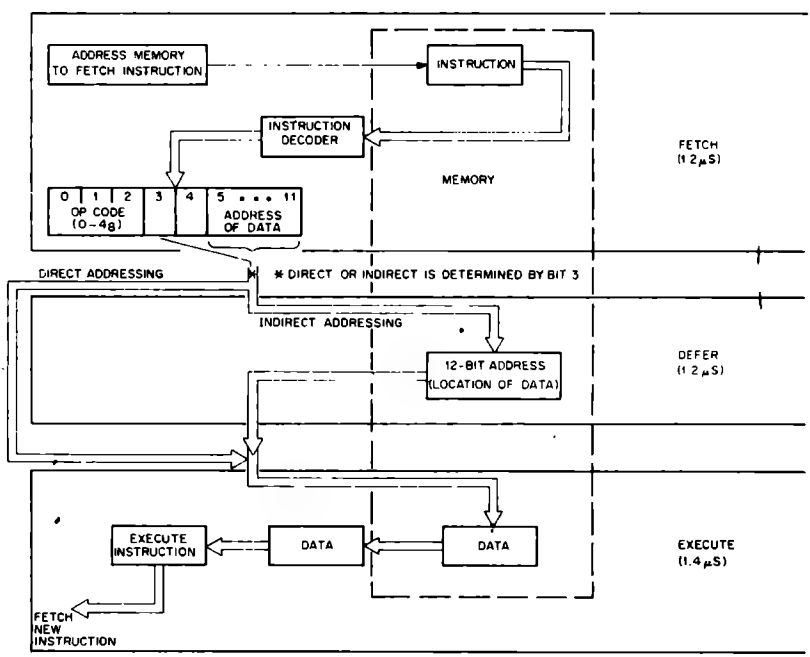


Figure 3-2 Memory Reference Instruction Simplified Flow

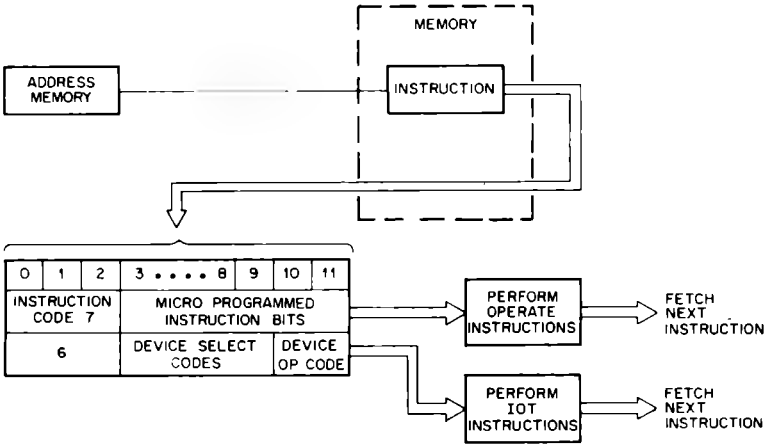


Figure 3-3 Augmented Instruction Simplified Flow Diagram

MEMORY REFERENCE INSTRUCTIONS

Logical AND (AND Y)

Octal Code: 0
Major States: F, (D), E
Execution Time: 2.6 μs with direct addressing, 3.8 μs with indirect addressing, 4.0 μs with auto-indexed indirect addressing.
Operation: The AND operation is performed between the contents of memory location Y and the contents of AC. The result is left in the AC, the original contents of the AC are lost, and the contents of Y are restored. Corresponding bits of the AC and Y are operated upon independently. This instruction, often called extract or mask, can be considered as a bit-by-bit multiplication.

Example:

Original ACJ	YJ	Final ACJ
0	0	0
0	1	0
1	0	0
1	1	1

Two's Complement Add (TAD Y)

Octal Code: 1
Major States: F, (D), E
Execution Time: 2.6 μs with direct addressing, 3.8 μs with indirect addressing, 4.0 μs with auto-indexed indirect addressing.

Operation: The contents of memory location Y are added to the contents of the AC in two's complement arithmetic. The result of this addition is held in the AC, the original contents of the AC are lost and the contents of Y are restored. If there is a carry from AC0, the link is complemented. This feature is useful in multiple precision arithmetic.

Increment and Skip if Zero (ISZ Y)

Octal Code: 2
Major States: F, (D), E
Execution Time: 2.6 μs with direct addressing, 3.8 μs with indirect addressing, 4.0 μs with auto-indexed indirect addressing.
Operation: The contents of memory location Y are incremented by one in two's complement arithmetic. If the resultant contents of Y equal zero, the contents of the PC are incremented by one and the next instruction is skipped. If the resultant contents of Y do not equal zero, the program proceeds to the next instruction. The incremented contents of Y are restored to memory. The contents of the AC are not affected by this instruction.

Deposit and Clear AC (DCA Y)

Octal Code: 3
Major States: F, (D), E
Execution time: 2.6 μs with direct addressing, 3.8 μs with indirect addressing, 4.0 μs with auto-indexed indirect addressing.
Operation: The contents of the AC are deposited in core memory at address Y and the AC is cleared. The previous contents of memory location Y are lost.

Jump to Subroutine (JMS Y)

Octal Code: 4
Major States: F, (D), E
Execution Time: 2.6 μs with direct addressing, 3.8 μs with indirect addressing, 4.0 μs with auto-indexed indirect addressing.
Operation: The contents of the PC are deposited in core memory location Y and the next instruction is taken from core memory location Y + 1. The contents of the AC are not affected by this instruction.

HOUSEKEEPING INSTRUCTION

Jump to Y (JMP Y)

Octal Code: 5
Major States: F, (D)
Execution Time: 1.2 μs with direct addressing, 2.4 μs with indirect addressing, 2.6 μs with auto-indexed indirect addressing.

Operation: Address Y is loaded into the PC so that the next instruction is taken from core memory address Y. The original contents of the PC are lost. The contents of the AC are not affected by this instruction.

AUGMENTED INSTRUCTIONS

Augmented instructions are one-cycle (FETCH) instructions that initiate various operations as a function of bit microprogramming. Augmented instructions are divided into two categories, neither of which are memory reference instructions. These are the INPUT/OUTPUT TRANSFER (IOT), which has an operation code of six; and the OPERATE, which has an operation code of seven. Bits 3 through 11 within each instruction function as an extension of the operations to be performed.

OPERATE INSTRUCTION

The operate instruction consists of three groups of microinstructions. Group 1 (OPR 1) is principally for clear, complement, rotate, and increment operations and is designated by the presence of a 0 in bit 3. Group 2 (OPR 2) is used principally in checking the contents of the accumulator and link and continuing to, or skipping, the next instruction based on the check. A 1 in bit 3 and a 0 in bit 11 designates an OPR 2 microinstruction. Group 3 is used to manipulate data between the MQ and AC Registers. A 1 in bits 3 and 11 designate an OPR 3 microinstruction. Each instruction is completed during a FETCH cycle. All operate instructions take place in 1.2 microseconds.

Group 1

The Group 1 microinstructions manipulate the contents of the accumulator and link. These instructions are microprogrammable; that is, they can be combined to perform specialized operations with other Group 1 instructions.

The Group 1 operate microinstruction format is shown in Figure 3-4, and the microinstructions are explained in the succeeding paragraphs. Bits within this group can be combined into one microinstruction. For example, it is possible to assign 1's to bits 5, 6, and 11, thereby creating an instruction which simultaneously clears the link, complements the accumulator and increments the accumulator. (The most frequently used combinations are listed in Appendix B).

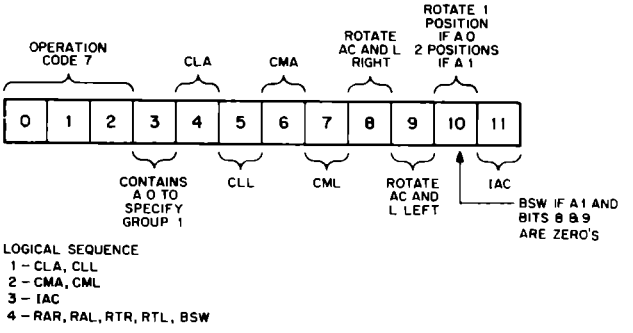

OPERATION CODE 7
CLA
CMA
ROTATE AC AND L RIGHT
ROTATE 1 POSITION IF A 0
2 POSITIONS IF A 1
0 1 2 3 4 5 6 7 8 9 10 11
CONTAINS A 0 TO SPECIFY GROUP 1
CLL
CML
ROTATE AC AND L LEFT
IAC
BSW IF A 1 AND BITS 8 & 9 ARE ZERO'S
LOGICAL SEQUENCE
1 – CLA, CLL
2 – CMA, CML
3 – IAC
4 – RAR, RAL, RTR, RTL, BSW

Figure 3-4 Group 1 Operate Instruction Bit Assignments

No Operation (NOP)

Octal Code: 7000
Sequence: None
Operation: This command causes a 1-cycle delay in the program before the next sequential instruction is initiated. This command is used to add execution time to a program, such as to synchronize subroutine or loop timing. The NOP also provides the programmer with a convenient means of removing an instruction.

Increment Accumulator (IAC)

Octal Code: 7001
Sequence: 3
Operation: The contents of the AC are incremented by one in two's complement arithmetic.

Rotate Accumulator Left (RAL)

Octal Code: 7004
Sequence: 4
Operation: The contents of the AC and link are rotated one binary position to the left. The contents of bits AC1-11 are shifted to the next greater significant bit, the content of AC0 is shifted into the L, and the content of the L is shifted into AC11.

Rotate Two Left (RTL)

Octal Code: 7006
Sequence: 4
Operation: The contents of the AC and link are rotated two binary positions to the left. This instruction is logically equal to two successive RAL operations.

Rotate Accumulator Right (RAR)

Octal Code: 7010
Sequence: 4
Operation: The contents of the AC and link are rotated one binary position to the right. The contents of bits AC0-10 are shifted to the next less significant bit, the content of AC11 is shifted into the L, and the content of the L is shifted into AC0.

Rotate Two Right (RTR)

Octal Code: 7012
Sequence: 4
Operation: The contents of the AC and link are rotated two binary positions to the right. This instrucion is logically equal to two successive RAR operations.

Byte Swap (BSW)

Octal Code: 7002
Sequence: 4
Operation: The right six bits of the accumulator are exchanged with the left six bits. AC0 is exchanged with AC6; AC1 with AC7, etc. The contents of the link are not affected.

Complement Link (CML)

Octal Code: 7020
Sequence: 2
Operation: The content of the L is complemented.

Complement Accumulator (CMA)

Octal Code: 7040
Sequence: 2
Operation: The contents of the AC are changed to the one's complement of the current contents of the AC. The content of each bit of the AC is complemented individually.

Complement and Increment Accumulator (CIA)

Octal Code: 7041
Sequence: 2, 3
Operation: The contents of the AC are converted from a binary value to their equivalent two's complement number. This conversion is accomplished by combining the CMA and IAC commands, thus the contents of the AC are complemented during sequence 2 and are incremented by one during sequence 3.

Clear Link (CLL)

Octal Code: 7100
Sequence: 1
Operation: The L is cleared (made equal to 0).

Set Link (STL)

Octal Code: 7120
Sequence: 1, 2
Operation: The L is set. This instruction is logically equal to combining the CLL and CML commands.

Clear Accumulator (CLA)

Octal Code: 7200
Sequence: 1
Operation: The content of each bit of the AC is cleared (made equal to 0).

Set Accumulator (STA)

Octal Code: 7240
Sequence: 1, 2
Operation: Each bit of the AC is set. This operation is logically equal to combining the CLA and CMA commands.

NOTE: The following codes are illegal and specifically reserved for future expansion: RAL RAR (octal code 7014), RTL RTR (octal code 7016) and any microprogramming of these bit combinations.

GROUP 2

Group 2 Operate microinstructions are often referred to as the "skip microinstructions" because they enable the programmer to perform tests on the accumulator and link and to skip the next instruction depending upon the results of the test. They are usually followed in a program by a JMP (or possibly a JMS) instruction. A skip instruction causes the computer to check for a specific condition, and if it is present, to skip the next instruction. If the condition were not present, the next instruction would be executed.

The Group 2 operate microinstruction format is shown in Figure 3-5 and the primary microinstructions are explained in the following paragraphs. Any logical combination of bits within this group can be composed into one microinstruction. (The instructions constructed by most logical command combinations are listed in Appendix B.)

If skips are combined in a single instruction the inclusive OR of the conditions determines the skip when bit 8 is a 0; and the AND of the inverse of the conditions determines the skip when bit 8 is a 1. For example, if 1s are designed in bits 6 and 7 (SZA and SNL), the next instruction is skipped if either the contents of the AC = 0, or the content of L= 1. If 1s are contained in bits 5, 7 and 8, the next instruction is skipped if the AC contains a positive number and the L contains a 0.

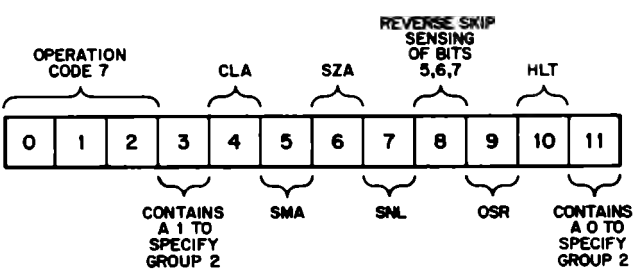

REVERSE SKIP SENSING OF BITS 5,6,7
OPERATION CODE 7
CLA
SZA
HLT
0 1 2 3 4 5 6 7 8 9 10 11
CONTAINS A 1 TO SPECIFY GROUP 2
SMA
SNL
OSR
CONTAINS A 0 TO SPECIFY GROUP 2
LOGICAL SEQUENCE:
1 (BIT 8 IS A 0) - EITHER SMA OR SZA OR SNL
(BIT 8 IS A 1) - BOTH SPA AND SNA AND SZL
2 - CLA
3 - OSR, HLT

Figure 3-5 Group 2 Operate Instruction Bit Assignments

Halt (HLT)

Octal Code: 7402
Sequence: 3
Operation: Clears the RUN flip-flop at Sequence 3, so that the program stops at the conclusion of the current machine

cycle. This command can be combined with others in the OPR 2 group that are executed during either sequence 1 or 2, and so are performed before the program stops.

OR with Switch Register (OSR)

Octal Code: 7404
Sequence: 3
Operation: The inclusive OR operation is performed between the contents of the AC and the contents of the SR. The result is left in the AC, the original contents of the AC are lost. The contents of the SR are unaffected by this command. When combined with the CLA command, the OSR performs a transfer of the contents of the SR into the AC.

Skip, Unconditional (SKP)

Octal Code: 7410
Sequence: 1
Operation: The contents of the PC are incremented by one so that the next sequential instruction is skipped.

Skip on Non-Zero Link (SNL)

Octal Code: 7420
Sequence: 1
Operation: The content of the L is sampled, and if it contains a 1, the contents of the PC are incremented by one so that the next sequential instruction is skipped. If the L contains a 0, no operation occurs and the next sequential instruction is initiated.

Skip on Zero Link (SZL)

Octal Code: 7430
Sequence: 1
Operation: The content of the L is sampled, and if it contains a 0 the contents of the PC are incremented by one so that the next sequential instruction is skipped. If the L contains a 1, no operation occurs and the next sequential instruction is initiated.

Skip on Zero Accumulator (SZA)

Octal Code: 7440
Sequence: 1
Operation: The content of each bit of the AC is sampled, and if all bits contain a 0 the contents of the PC are incremented by one so that the next sequential instruction is skipped. If any bit of the AC contains a 1, no operation occurs and the next sequential instruction is initiated.

Skip on Non-Zero Accumulator (SNA)

Octal Code: 7450
Sequence: 1
Operation: The content of each bit of the AC is sampled, and if any bit contains a 1 the contents of the PC are incremented by one so that the next sequential instruction is skipped. If all bits of the AC contain a 0, no operation occurs and the program control advances to the next sequential instruction.

Skip on Minus Accumulator (SMA)

Octal Code: 7500
Sequence: 1
Operation: The content of the most significant bit of the AC is sampled, and if it contains a 1, indicating that the AC contains a negative two's complement number, the contents of the PC are incremented by one so that the next sequential instruction is skipped. If the AC contains a positive number no operation occurs and program control advances to the next sequential instruction.

Skip on Positive Accumulator (SPA)

Octal Code: 7510
Sequence: 1
Operation: The content of the most significant bit of the AC is is sampled, and if it contains a 0, indicating a positive (or zero) two's complement number, the contents of the PC are incremented by one so that the next sequential instruction is skipped. If the AC contains a negative number, no operation occurs and program control advances to the next sequential instruction.

Clear Accumulator (CLA)

Octal Code: 7600
Sequence: 2
Operation: Each bit of the AC is cleared to contain a binary 0.

Group 3

The Group 3 Operate microinstructions are concerned with the manipulation of data between the AC and the MQ registers. The MQ register is an auxiliary register for the temporary storage of data. It is convenient to store data in an auxiliary register without having to go into memory. Group 3 instructions allow the loading, swapping of AC and the MQ register contents, inclusive ORing of the MQ with AC registers and clearing both registers. Although the register and instructions are primarily intended to be used with the Extended Arithmetic KE8-E option, they are available for use as a standard feature.

The format of the Group 3 instructions is shown in Figure 3-6. Having an operation code of 7, this instruction class is identified as group 3 only when both bits 3 and 11 contain a 1.

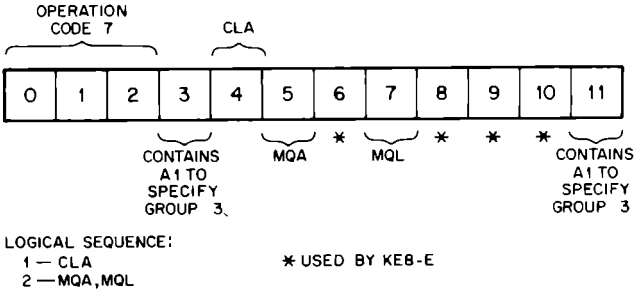


Figure 3-6 Bit Assignments for Group 3 Operate Instructions

Bits 6, 8, 9 and 10 should be zero when the KE8-E option is not employed. The description of the four Group 3 instructions is given in the following:

Instructions

Clear Accumulator and Multiplier Quotient (CAM)

Octal Code: 7621
Execution Time: 1.2 μs
Operation: Clears the AC during logical sequence 1, as in CLA; during logical sequence 2, the MQ is cleared.

Multiplier Quotient Load into Accumulator (MQA)

Octal Code: 7501
Execution Time: 1.2 μs
Operation: The contents of the MQ are inclusively ORed with the AC, and the result loaded into the AC. The previous contents of the AC are lost, but the contents of the MQ are not affected. This instruction provides the programmer with a direct inclusive OR instruction. This instruction may also be combined with a CLA instruction to effect a direct transfer of information from MQ to AC.

Load Multiplier Quotient (MQL)

Octal Code: 7421
Execution Time: 1.2 μs
Operation: Loads the content of the AC into the MQ, and then clears the AC.

Swap MQ and AC (SWP)

Octal Code: 7521
Execution Time: 1.2 μs
Operation: The contents of AC and MQ are exchanged. This instruction can be combined with a CLA instruction

to move the contents of MQ to AC and then clear the MQ.

INPUT OUTPUT TRANSFER (IOT)

Octal Code: 6
Major State: F
Execution Time: If the selected device is internal, the IOT takes place in 1.2 microseconds.
If the selected device is external, the computer enters an expanded cycle of 2.6 microseconds (if the IOP ends in 1, 2 or 4); 3.6 microseconds (if the IOP ends in 3, 5, or 6); or 4.6 microseconds (if the IOP ends in 7). An IOT ending in 0 always takes place in 1.2 microseconds.

Operation: Input/output transfer (IOT) instructions initiate the operation of peripheral equipment and effect information transfers between the processor and an I/O device. Upon recognition of the operation code 6 as an IOT instruction, the computer determines whether the selected device is internal (plugged directly into the OMNIBUS) or external (connected via the KA8-E Positive I/O Bus Interface module). The nature of the OMNIBUS is such that IOT's such as 6000 can be used for control codes for devices directly connected to the OMNIBUS, since the last 3 bits are decoded to determine the device operation.

The last 3 bits of the instruction cause the generation of "IOP PULSES" at the External Bus Interface as follows:

Instruction Bit	IOP	Sequence Time
11	IOP 1	1
10	IOP 2	2
9	IOP 4	3

IOP pulses enact a data transfer or initiate a control operation. Selection of an equipment is accomplished by bits 3 through 8 of the IOT instruction. These bits form a 6-bit code that enables the device selector in a given device.

The format of the IOT instruction is shown in Figure 3-7. Operations performed by IOT microinstructions are explained in Chapters 5, 9, and 10.

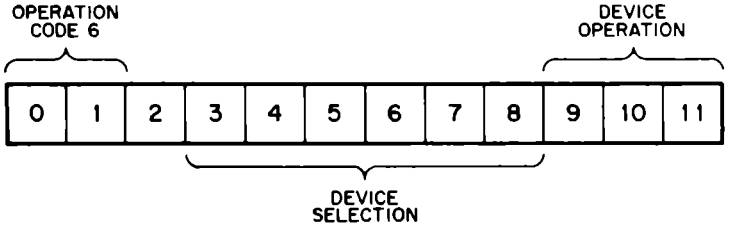


Figure. 3-7 IOT Instruction Bit Assignment

PROGRAM INTERRUPT

The program interrupt features allow certain external conditions to interrupt the computer program. Program interrupts are used to either speed the information processing of input/output devices or allow certain alarms to halt the program in progress and initiate another routine. When a program interrupt request is made, the computer completes execution of the instruction in progress before acknowledging the request and entering the interrupt mode. A program interrupt is similar to a JMS to location 0; that is, the contents of the program counter are stored in location 0, and the program resumes operation in location 1 with the interrupt disabled. The interrupt program commencing in location 1 is responsible for identifying the signal causing the interruption, for removing the interrupt condition, and for returning to the original program with the interrupt re-enabled. Exit from the interrupt program, back to the original program, can be accomplished by a JMP I 0 instruction.

When an interrupt request is acknowledged, the interrupt is automatically disabled by the program interrupt synchronization circuits (not by instructions). The next instruction is taken from core memory location 1. Usually, the instruction stored in locations 1 is a JMP, which transfers program control to a subroutine which services the interrupt. At some time during this subroutine, an ION instruction must be given. The ION can be given at the end of the subroutine, just before control is transferred back to the original program. In this application, the ION instruction immediately precedes the last instruction in the routine. A delay of one instruction (regardless of the execution time of the following instruction), is inherent in the ION instruction to allow transfer of program control back to the original program before enabling the interrupt. Exit from the subroutine usually is accomplished by a JMP I 0 instruction.

The ION command can also be given during the subroutine as soon as it has determined the I/O device causing the interrupt. This latter method allows the subroutine which is handling a low priority interrupt to be interrupted, possible by a high priority device. Programming of an interrupt subroutine, which checks for priority and allows itself to be interrupted, must make provisions to relocate the contents of the program counter stored in location 0; so that the return address to the original program is not lost if another interrupt occurs.

Instructions

Interrupt Turn On (ION)

Octal Code: 6001

Operation: This command enables the computer to respond to a program interrupt request. If the interrupt is disabled when this instruction is given, the computer executes the next instruction, then enables the interrupt. The additional instruction allows exit from the interrupt subroutine before allowing another interrupt to occur. This instruction has no effect upon the condition of the interrupt circuits if it is given when the interrupt is enabled.

Skip If Interrupt ON (SKON)

Octal Code: 6000

Operation: The state of the interrupt enable flip-flop is tested. If this flip-flop is set, the next sequential instruction is skipped. Simultaneously with this test (and before a device flag can cause an interrupt) the interrupt system is turned off as described under the following IOF instruction.

Skip If Interrupt Request (SRQ)

Octal Code: 6003

Operation: The state of the internal interrupt request bus is tested. If it is low, indicating one or more devices are requesting an interrupt, the next instruction is skipped.

Interrupt Turn Off (IOF)

Octal Code: 6002

Operation: This command disables the program interrupt synchronization element to prevent interruption of the current program.

Flag Processing IOTs

Get Flags (GTF)

Operation: 6004

Octal Code: The following machine states are read into the indicated bits of the accumulator:

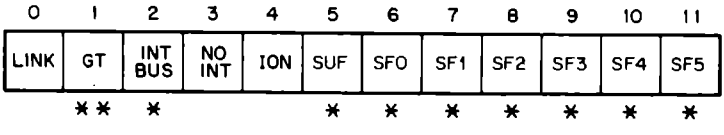


Figure 3-8 Flag Processing States

* Only if Extended Memory Control, type KM8-E, installed.

† Only if Extended Arithmetic Element (EAE), type KE8-E, installed.

Restore Flags (RTF)

Octal Code: 6005
Operation: The contents of the AC are loaded into the flags. This instruction is the converse of GTF. The interrupt is enabled, regardless of the state of bit 4. Bits 6-8 and 9-11 are loaded into the instruction buffer (IB) and data field (DF) of the KM8-E. Bit 2 is ignored.

Skip If "Greater than" Flag Is Set (SGT)

Octal Code: 6006
Operation: If the GT flag is set, the next instruction is skipped. This instruction is implemented only if the KE8-E is installed.

Clear all Flags (CAF)

Octal Code: 6007
Operation: This instruction is logically equivalent to operating the CLEAR key on the panel. It generates INITIALIZE on the OMNIBUS and at the external I/O interface. The LINK and AC are cleared. The action of INITIALIZE is a function of the design of each peripheral, but generally INITIALIZE clears flags and motion control flip-flops, and sets the interrupt enable flip-flop of all peripherals.

INSTRUCTION SUMMARY

A summary of the common usage of the eight basic instructions is provided in Table 3-1. As the reader continues on with the remaining chapters in this handbook, he should keep in mind the capability of all eight instructions since the processor's operation is determined by the use of each instruction. A program, for instance, is simply a collection of instructions strung out in a particular order for a particular purpose such as solving a problem. Each instruction performs a series of steps to satisfy the requirements of the program.

Table 3-1

Basic Instruction Usage Summary

BASIC INSTRUCTION	MINIMUM NO. OF CYCLES	COMMON USAGE	EXAMPLE OF USAGE
AND (0)(octal)	2	Data Manipulation	Strips unwanted bits from the AC.
TAD (1)(octal)	2	Data Manipulation	Provides arithmetic addition. Also serves to load the AC with the contents of some memory location.

Table 3-1 (Cont.)

Basic Instruction Usage Summary

BASIC INSTRUCTION	MINIMUM NO. OF CYCLES	COMMON USAGE	EXAMPLE OF USAGE
ISZ (2)(octal)	2	Program Loops	Used for counting.
DCA (3)(octal)	2	Data Manipulation	Used when placing data into some memory location.
JMS (4)(octal)	2	Subroutine Entry	Provides entry to subroutines.
JMP (5)(octal)	1	Manipulation of Program Counter	Allows the programmer to go to a different portion of his program. Also provides subroutine exit.
IOT (6)(octal)	1	External Communication	Allows the program to converse with peripherals.
OPR (7)(octal)	1	Testing of AC and Link	Operate and/or test the contents of the AC and Link. Operate on the contents of the MQ Register.

CHAPTER 4

MEMORY AND PROCESSOR BASIC PROGRAMMING

GENERAL

This chapter deals with the concepts required to program the PDP-8/E and identifies the system programs available to the user. Two handbooks, INTRODUCTION TO PROGRAMMING and PROGRAMMING LANGUAGES, provide a more detailed treatment and description of the commonly used programming languages and programming systems.

Organization of the standard core memory or any 4096-word field of extended memory is summarized as follows:

Total locations (decimal)	0-4095 or 4096
Total addresses (octal)	0-7777 or 10,000
Number of pages (decimal)	0-31 or 32
Page designations (octal)	0-37 or 40
Number of locations per page (decimal)	0-127 or 128
Addresses within a page (octal)	0-177 or 200

Routines using 128 instructions or less can be written in one page using direct addresses for looping and indirect addresses for data stored in other pages. When planning the location of instructions and data in core memory, the following locations are reserved for special purposes:

Address	Purpose
0 (octal)	Stores the contents of the program counter following a program interrupt.
1 (octal)	Stores the first instruction to be executed following a program interrupt.

10 (octal)—17 (octal) Auto-indexing.

MEMORY ADDRESSING

The programmer has 4096 (decimal) locations which he may address. However, as illustrated in Figure 3-2 of Chapter 3, when an instruction is fetched from memory, only bits 5 through 11 contain the address of the data. Addressing is accomplished using octal notation. Therefore the 4096 possible locations require addresses in octal from 0000 to 7777. This means that a total of 12 bits is required to specify an absolute address. So that all locations may be addressed as efficiently as possible, memory is addressed in terms of pages with a coding scheme that allows easy access to any one of the 10,000 octal locations. The page addressing scheme is illustrated in Figure 4-1 which shows the relationship of the 40 octal pages with the 10,000 octal locations. The programmer is interested in only three pages in memory at any one time:

a. The current page
b. Page 0
c. A location on other than the current page or page 0.

Page 0 is used to store commonly used operands and off-page pointers. For instance, the location of an indirect address used by instructions is usually Page 0.

ABSOLUTE ADDR (OCTAL)	CORE MEMORY PAGE (OCTAL)	PAGE ADDR (OCTAL)
7777 7600	37	177 000
7577 7400	36	177 000
7377 7200	35	177 000
7177 7000	34	177 000
6777 6600	33	177 000
6577 6400	32	177 000
6377 6200	31	177 000
6177 6000	30	177 000
5777 5600	27	177 000
5577 5400	26	177 000
5377 5200	25	177 000
5177 5000	24	177 000
4777 4600	23	177 000
4577 4400	22	177 000
4377 4200	21	177 000
4177 4000	20	177 000
3777 3600	17	177 000
3577 3400	16	177 000
3377 3200	15	177 000
3177 3000	14	177 000
2777 2200	13	177 000
2577 2000	12	177 000
2377 1600	11	177 000
2177 1400	10	177 000
1777 1600	7	177 000
1577 1400	6	177 000
1377 1200	5	177 000
1177 1000	4	177 000
0700 0600	3	177 000
0577 0400	2	177 000
0377 0200	1	177 000
0177 0000	0	177 000

Figure 4-1 Memory Addressing Scheme

The format of the Central Processor (CPMA) Memory Address Register establishing the memory page must first be considered. The MA register is an unequally divided 12-bit register in which the least significant bits of the MA (bits 5-11) are called the page address bits and the most significant bits (bits 0-4) are called the page bits. The 12 bits of the Memory Address are established by the program counter (PC) and are re-established each time the PC loads an absolute address for the next instruction. Bits 0-4 are used to establish the memory page as shown in Figure 4-2. Because the pages to be addressed include pages 0-37 (octal), only five bits are required. The first two bits represent numbers from 0 to 3 and the next three represent numbers from 0 to 7. Because the locations to be addressed on any given page include locations 0-177 (octal), only seven bits are required to specify any one location. Bit 5 represents an octal 1 or 0; bits 6, 7, and 8 represent the second octal digit from 0 to 7; bits 9, 10, and 11 represent the last octal digit from 0 to 7. Thus, on the example shown in Figure 4-2, the MA register is addressing page 5, location 73 (absolute address 1273).

When the user first receives his PDP-8/E, he should assume that he has a processor containing no information content in memory. Before he can load instructions into memory, he must first perform the initializing and loading procedures described in Chapter 2. The following discussion assumes that the preliminary procedures have been completed and that the programmer now wants to load into core those instructions which will be called upon after a program has been written. His main concern is to decide in which memory locations he desires to place his instructions and in which locations he wishes to place the corresponding data.

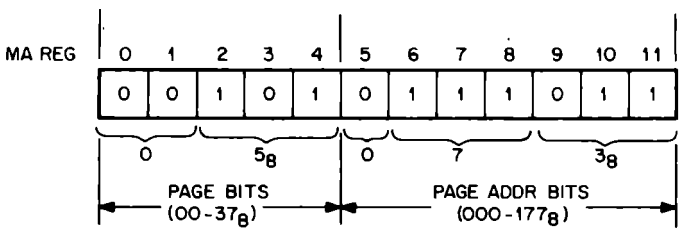


Figure 4-2 Format Establishing Memory Page

Initially, the programmer must load the Central Processor Memory Address Register with an address and then deposit a 12-bit Memory Reference Instruction word in the format shown in Figure 4-3.

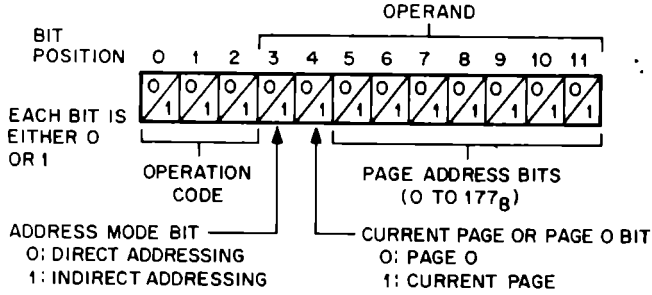

OPERAND
BIT POSITION 0 1 2 3 4 5 6 7 8 9 10 11
EACH BIT IS EITHER 0 OR 1
OPERATION CODE
PAGE ADDRESS BITS (0 TO 177₈)
ADDRESS MODE BIT
0: DIRECT ADDRESSING
1: INDIRECT ADDRESSING
CURRENT PAGE OR PAGE 0 BIT
0: PAGE 0
1: CURRENT PAGE

Figure 4-3 Format of a Memory Reference Instruction

The first three bits contain the instruction operation code and have nothing to do with addressing. The last seven bits address the location (from 0 to 177 octal), which will contain either data or a 12-bit address. Those address bits are located in the Memory Buffer Register, and are ineffective until bits 3 and 4 are decoded, at which time the address bits are transferred from the Memory Buffer to the Central Processor Memory Address Register. The page address (the first five bits of the MA register) is determined by whether bit 3 is a 0 or 1 (see Figure 4-4).

Where to place the instruction word or Data word is a very important consideration. At this point, the programmer has three choices in the location of the data:

a. the current page (that page containing the Instruction)
b. page 0
c. a page other than page 0 or the current page.

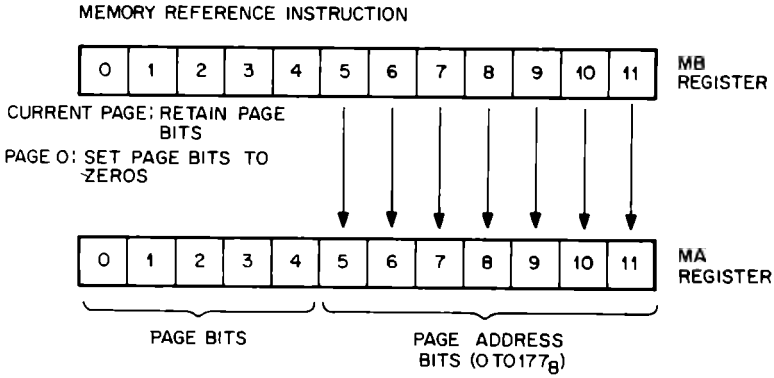

MEMORY REFERENCE INSTRUCTION
0 1 2 3 4 5 6 7 8 9 10 11
MB REGISTER
CURRENT PAGE: RETAIN PAGE BITS
PAGE 0: SET PAGE BITS TO ZEROS
0 1 2 3 4 5 6 7 8 9 10 11
MA REGISTER
PAGE BITS
PAGE ADDRESS BITS (0 TO 177₈)

Figure 4-4 Transfer of Address Portion of MRI into MA Register

Current Page—If the programmer desires the data to be located in the current page, he must make bit 3 a 1 in the original instruction word. The logic within the processor causes the first five bits of the MA to remain, and transfers the last seven bits of the MB (the new address within a page) to the last seven bits of the MA Register. This method of updating the MA Register is illustrated in Figure 4-4.

Page 0—Page 0 is commonly used to store operands or an indirect address of the data. The programmer must set bit 3 of the original instruction word to 0. The logic within the processor then places all zeros in MA bits 0 through 4 and transfers the content of the last 7 bits of the MB register to the last 7 bits of the MA Register. Thus, the page address is now page 0 and the address within page 0 is some address between 0 and 177 octal. This is illustrated in Figure 4-4.

Addressing A Page Other Than the Current Page or Page 0—The programmer may address a page other than the current page or page 0 by placing a 1 in bit 3 of the original instruction word. As before, the computer then goes to an address on the current page or on page 0, depending on the state of bit 4. The logic within the processor responds to bit 3 being a 1 by going into a defer state for a new address. This procedure is called "Indirect Addressing."

INDIRECT ADDRESSING

In the preceding section, the method of directly addressing 400(octal) memory locations by an MRI was described—namely those on page 0 and those on the current page. This section describes the method for addressing the other 7400(octal) memory locations. Bit 3 of an MRI designates the address mode. When bit 3 is a 0, the operand is a direct address. When bit 3 is a 1, the operand is an indirect address. An indirect address (pointer address) identifies the location that contains the desired address (effective address). To address a location that is not directly addressable, the absolute address of the desired location is stored in one of the 400(octal) directly addressable locations (pointer address); the pointer address is written as the operand of the MRI; and the letter I is written between the mnemonic and the operand. (During assembly, the presence of the I results in bit 3 of the MRI being set to 1.) Upon execution, the MRI will operate on the contents of the location identified by the address contained in the pointer location.

The two examples in Figure 4-5 illustrate the difference between direct addressing and indirect addressing. The first example shows a TAD instruction that uses direct addressing to get data stored on page 0 in location 50; the second is a TAD instruction that uses indirect addressing, with a pointer on page 0 in location 50, to obtain data stored in location 1275. (When references are made to them from various pages, constants and pointer addresses can be stored on page 0 to avoid the necessity of storing them on each applicable page.) The octal value 1050, in the first example, represents direct addressing (bit 3 = 0); the octal value 1450, in the second example, represents indirect addressing (bit 3 = 1). Both examples assume that the accumulator has previously been cleared.

Location	Content	
200	TAD 50	(TAD 50 = 1050_8)
.	.	Instruction
.	.	
50	1275	Data (Number) To Be Acted Upon By
.	.	Instruction Address
.	.	
1275	20	(Content of location 1275 is not used in the execution of the instruction in location 200.)

NOTE: AC = 1275 after executing the instruction in location 200

Location	Content	
200	TAD I 50	(TAD I 50 = 1450_8)
.	.	
.	.	Designates Indirect Addressing
.	.	Instruction
50	1275	Pointer Address
.	.	
.	.	
1275	20	Data (Number) To Be Acted Upon By Instruction Effective Address

NOTE: AC = 20 after executing the instruction in location 200.

Figure 4-5 Comparison of Direct and Indirect Addressing

The following three examples illustrate some additional ways in which indirect addressing can be used. As shown in example 1, indirect addressing makes it possible to transfer program control off page 0 (to any desired memory location). (Similarly, indirect addressing makes it possible for other memory reference instructions to address any of the 4,096(10) memory locations.) Example 2 shows a DCA instruction that uses indirect addressing with a pointer on the current page. The pointer in this case designates a location off the current page (location 227) in which the data is to be stored. (A pointer address is normally stored on the current page when all references to the designated location are from the current page.) Indirect addressing provides the means for returning to a main program from a subroutine, as shown in example 3. Indirect addressing is also effectively used in manipulating tables of data.

EXAMPLE 1

Location	Content	
75	JMP I 100	(JMP I 100 = 5500(octal))
.	.	
.	.	Designates Indirect Addressing Instruction

100	6000	Pointer Address
6000	DCA 6100	Next Instruction To Be Executed

NOTE: Execution of the instruction in location 75 causes program control to be transferred to location 6000, and the next instruction to be executed is the DCA 6100 instruction.

EXAMPLE 2

Location	Content	
450	DCA I 577	(DCA I 577 = 3777(octal)) Designates Indirect Addressing Instruction
577	277	Pointer Address
227	nnnn	Data (Number) Stored By Instruction (Effective Address)

NOTE: Execution of the instruction in location 450 causes the contents of the accumulator to be stored in location 227.

EXAMPLE 3

Location	Content	
207	JMS I 70	(JMS I 70 = 4470(octal))
210	TAD 250	(The next instruction to be executed upon return from the subroutine.)
70	2000	(Starting address of the subroutine stored here.)
2000	aaaa	(Return address stored here by JMS instruction.)
2001	iii	(First instruction of subroutine.)
2077	JMP I 2000	(Last instruction of subroutine.)

NOTES: 1. Execution of the instruction in location 207 causes the address 210 to be stored in location 2000 and the instruction in location 2001 to be executed next. Execution of the subroutine proceeds until the last instruction (JMP I 2000) causes control to be transferred back to the main program, continuing with the execution of the instruction stored in location 210.

2. A JMS instruction that uses indirect addressing is useful when the subroutine is too large to store on the current page.
3. Storing the pointer address on page 0 enables instructions on various pages to have access to the subroutine.

PROGRAMMING OPERATIONS

The programmer can make use of any combination of instructions. The following sections describe the more common programming operations.

STORING AND LOADING

Data is stored in any core memory location by use of the DCA (Deposit & Clear AC) instruction. This instruction clears the AC to simplify loading of the next data. If the data deposited is required in the AC for the next program operation, the DCA must be followed by a TAD for the same address. All loading of core memory information into the AC is accomplished by means of the TAD instruction.

The DCA instruction stores the contents of the AC in the referenced location, destroying the original contents of the location. The AC is then set to all zeroes. The following example shows the contents of the accumulator, link, and location 225 before and after executing the instruction DCA 225.

DCA 225

	AC	Link	Loc. 225
Before Execution	1234	1	7654
After Execution	0000	1	1234

The following facts should be kept in mind when using the DCA instruction:

a. The state of the link bit is not altered.
b. The AC is cleared.
c. The original contents of the addressed location are replaced by the value of the AC.

PROGRAM CONTROL

The Program Counter is used to direct the processor to the next address in a sequence of addresses. Therefore, the content of the PC Register is always one more than the content of the Central Processor Memory Address (CPMA) Register. When an instruction has been completed and the processor is ready to go into a new fetch, the content of the Program Counter is transferred into the CPMA Register and the Program Counter with its original address is incremented by +1, thereby pointing to the next sequential address. This procedure is called Program Control because it directs the processor to the next instruction. Because this rigid sequence of instructions is not always desirable for practical programming, the PDP-8/E provides a means of jumping out of this sequence to transfer Program Control from one sequence of instructions to another or to allow the processor to enter a subroutine which is itself a sequence of instructions and re-enter the main program when the subroutine has been completed.

Transfer of program control to any core memory location uses the JMP or JMS instructions. The JMP I (indirect address, bit 3 = 1) is used to transfer program control to any location in core memory which is not in the current page or page 0.

The JMS Y is used to enter a subroutine which starts at location Y +1 in the current page or page 0. The contents of the PC are stored in the specified address Y, and address Y + 1 is transferred into the PC. To exit a subroutine, the last instruction is a JMP I Y, which returns program control to the location stored in Y.

The JMP instruction loads the effective address of the instruction into the program counter, thereby changing the program sequence since the PC specifies the next instruction to be performed. In the following example, execution of the instruction in location 250 (JMP 300) causes the program to jump over the instructions in locations 251 through 277 and immediately transfer control to the instruction in location 300.

Location	Content	
250	JMP 300	(This instruction transfers program control to location 300.)
.	.	
.	.	
300	DCA 300	
.	.	

NOTE: The JMP instruction does not affect the contents of the AC or link.

A program written to perform a specific operation often includes sets of instructions which perform intermediate tasks. These intermediate tasks may be finding a square root, or typing a character on a keyboard. Such operations are often performed many times in the running of one program and may be coded as subroutines. To eliminate the need of writing the complete set of instructions each time the operation must be performed, the JMS (jump to subroutine) instruction is used. The JMS instruction stores a pointer address in the first location of the subroutine and transfers control to the second location of the subroutine. After the subroutine is executed, the pointer address identifies the next instruction to be executed. Thus, the programmer has at his disposal a simple means of exiting from the normal flow of his program to perform an intermediate task and a means of returning to the correct location upon completion of the task. (This return is accomplished using indirect addressing, which is discussed elsewhere in this chapter.)

The following example illustrates the action of the JMS instruction:

Location	Content	
PROGRAM		
200	JMS 350	(This instruction stores 0201 in location 350 and transfers program control to location 351.)

```
201         DCA 270     (This instruction stores the contents
                        of the AC in location 270 upon return
                        from the subroutine.)
 .           .
 .           .
 .           .
SUBROUTINE
350         0000        (This location is assumed to have an
                        initial value of 0000; after JMS 350
                        is executed, it is 0201.)
351         i i i       (First instruction of subroutine)
 .           .
 .           .
 .           .
375         JMP I 350   (Last instruction of subroutine)
```

The following should be kept in mind when using the JMS:

1. The value of the PC (the address of the JMS instruction +1) is always stored in the first location of the subroutine, replacing the original contents.
2. Program control is always transferred to the location designated by the operand + 1 (second location of the subroutine).
3. The normal return from a subroutine is made by using an indirect JMP to the first location of the subroutine (JMP I 350 in the above example); (Indirect addressing, as discussed in this chapter effectively transfers control to location 201).
4. When the results of the subroutine processing are contained in the AC and are to be used in the main program, they must be stored upon return from the subroutine before further calculations are performed. (In the above example, the results of the subroutine processing are stored in location 270.)

ARITHMETIC OPERATIONS

One arithmetic instruction is included in the order code, the two's complement add (TAD). Using this instruction, routines can easily be written to perform addition, subtraction, multiplication, and division in two's complement arithmetic.

Two's Complement Arithmetic

In two's complement arithmetic, addition, subtraction, multiplication, and division of binary numbers are performed in accordance with the common rules of binary arithmetic. In the PDP-8/E, as in other machines utilizing complementation techniques, negative numbers are represented as the complements of positive numbers, and subtraction is achieved by complement addition. Representation of negative values in one's complement arithmetic is slightly different from that in two's complement arithmetic.

The one's complement of a number is the complement of the absolute positive value; that is, all 1s are replaced by 0s and all 0s are replaced by 1s. The two's complement of a number is equal to the one's complement of the positive value plus one.

In one's complement arithmetic a carry from the sign bit (most significant bit) is added to the least significant bit in an end-around carry. In two's complement arithmetic a carry from the sign bit complements the link (a carry would set the link to 1 if it were properly cleared before the operation), and there is no end-around carry.

The TAD instruction (see Figure 4-6) performs a binary addition between the specified data word and the contents of the accumulator, leaving the result of the addition in the accumulator. If a carry out of the most significant bit of the accumulator should occur, the state of the link bit is complemented. The add instruction is called a Two's Complement Add to remind the programmer that negative numbers must be expressed as the two's complement of the positive value.

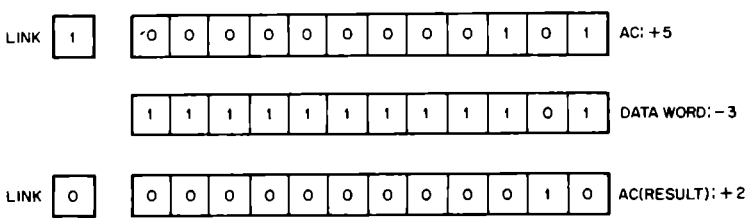


Figure 4-6 Operation of the TAD Instruction

The following points should be remembered when using the TAD instruction:

a. Negative numbers must be expressed as a two's complement of the positive value of the number.
b. A carry out of the accumulator will complement the link.
c. The data word in the referenced location is not affected.

LOGIC OPERATIONS

The PDP-8/E instruction list includes the logic instruction AND. A short routine can be written from this instruction to perform the exclusive OR operation.

Logical AND

The logical AND operation between the contents of the accumulator and the contents of a core memory location Y is performed directly by means of the AND Y instruction. The logical AND performs an AND operation with AC and MB, AC1 and MB1, etc. The result remains in the AC, the original contents of the AC are lost, and the contents of location Y are unaffected.

The AND instruction causes a bit-by-bit Boolean AND operation between the contents of the accumulator and the data word specified by the instruction. The result is left in the accumulator as shown in Figure 4-7.

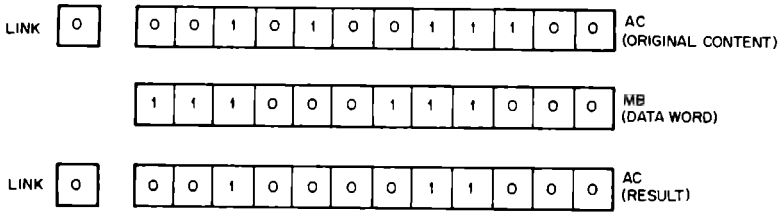


Figure 4-7 Operation of the AND Instruction

The following points should be noted with respect to the AND instruction:

a. A 1 appears in the AC only when a 1 is present in both the AC and the data word (The data word is often referred to as a mask).
b. The state of the link bit is not affected by the AND instruction.
c. The data word in the referenced location is not altered.

Inclusive OR

The Inclusive OR instruction makes use of the MQ register, which is a permanent part of the PDP-8/E. Assuming that value A is in the AC and value B is stored in a known core memory address, the following sequence performs the inclusive OR. The sequence is stated as a utility subroutine called IOR.

```
/CALLING SEQUENCE                JMS IOR
/                                (ADDRESS OF B)
/ENTER WITH ARGUMENT IN AC; EXIT WITH
/LOGICAL RESULT IN AC

/ADDRESS LABEL INSTRUCTION              REMARKS
    IOR,       0
               MQL              /AC TO MQ, CLEAR AC
               TAD I IOR        /GET ADDRESS OF SECOND
               DCA TEM1         ARGUMENT
               TAD I TEM1
               MQA              /IOR MQ TO AC
               ISZ IOR
               JMP I IOR
  TEM1,        0
```

Exclusive OR

The exclusive OR operation for two numbers, A and B, can be performed by a subroutine called by the mnemonic code XOR. In the following general purpose XOR subroutine, the value A is assumed to be in the AC, and the address of the value B is assumed to be stored in a known core memory location.

```
/CALLING SEQUENCE        JMS XOR
/                        (ADDRESS OF B)
/                        (RETURN)
      /ENTER WITH ARGUMENT IN AC; EXIT WITH
            /LOGICAL RESULT IN AC
```

```
XOR,    0
        DCA TEM1
        TAD I XOR
        DCA TEM2
        TAD TEM1
        AND I TEM2
        CMA IAC
        CLL RAL
        TAD TEM1
        TAD I TEM2
        ISZ XOR
        JMP I XOR
TEM1,   0
TEM2,   0
```

An XOR subroutine can be written using fewer core memory locations by making use of the IOR subroutine; however, such a subroutine takes more time to execute. A faster XOR subroutine can be written by storing the value B instead of the address of B, in the second instruction of the calling sequence; however, the resulting subroutine is not as useful as the subroutine given here.

INDEXING OPERATIONS

External events can be counted by the program, and the number can be stored in core memory. The core memory location used to store the event count can be initialized (cleared) by a CLA command followed by a DCA instruction. Each time the event occurs, the event count can be advanced by a sequence of commands such as CLA, TAD, IAC, and DCA.

The ISZ instruction is used to count repetitive program operations or external events without disturbing the contents of the accumulator. Counting a specified number of operations is performed by storing a two's complement negative number equivalent to the number of operations to be counted. Each time the operation is performed, the ISZ instruction is used to increment the contents of this stored number and to check the result. When the stored number becomes zero, the specified number of operations have occurred and the program skips out of the loop and back to the main sequence.

This instruction is also used for other routines in which the contents of a memory location are incremented without disturbing the contents of the accumulator, such as storing information from an I/O device in sequential memory locations, or using core memory locations to count I/O device events.

The ISZ instruction adds a 1 to the referenced data word and then examines the result of the addition. If a zero result occurs, the instruction following the ISZ is skipped. If the result is not zero, the instruction following the ISZ is performed. In either case, the result of the addition replaces the original data word in memory. The example below illustrates one method of adding the contents of a given location to the AC a specified number of times (multiplying) by using an ISZ instruction to increment a tally. The effect of this example is to multiply the contents of location 275 by 2. (To add the contents of a given location to the AC

twice, using the ISZ loop, as shown below, requires more instructions than merely repeating the TAD instruction. However, when adding the contents four or more times, use of the ISZ loop requires fewer instructions.) In the first pass of the example, execution of ISZ 250 increments the contents of location 250 from 7776 to 7777 and then transfers control to the following instruction (JMP 200). In the second pass, execution of ISZ 250 increments the contents of location 250 from 7777 to 0000 and transfers control to the instruction in location 203, skipping over location 202.

CODING FOR ISZ LOOP

Location	Content
200	TAD 275
201	ISZ 250
202	JMP 200
203	DCA 276
.	.
.	.
.	.
250	7776
.	.
.	.
.	.
275	0100
276	0000

SEQUENCE OF EXECUTION FOR ISZ LOOP

		Content After Instruction Execution			
Location	Content	AC	250	275	276
FIRST PASS					
200	TAD 275	0100	7776	0100	0000
201	ISZ 250	0100	7777	0100	0000
202	JMP 200	0100	7777	0100	0000
SECOND PASS					
200	TAD 275	0200	7777	0100	0000
201	ISZ 250	0200	0000	0100	0000
202	JMP 200	(Skipped	during	second	pass)
203	DCA 276	0000	0000	0100	0200

ISZ Instruction Incrementing a Tally

The following points should be kept in mind when using the ISZ instruction:

a. The contents of the AC and link are not disturbed.
b. The original word is replaced in main memory by the incremented value.
c. When using the ISZ for looping a specified number of times, the tally must be set to the negative of the desired number.
d. The ISZ performs the incrementation first and then checks for a zero result.

CODING A PROGRAM

The introduction of an assembler in Chapter 2 enabled the programmer to write a symbolic program using meaningful mnemonic codes rather than the octal representation of the instructions. The programmer could now write mnemonic programs such as the following example, which multiplies 18(10) by 36(10) using successive addition.

```
200     CLA CLL     Initialize
201     TAD 210     Set up a Tally
202     CIA         equal to -18(10) to
203     DCA 212     count the additions of 36
204     TAD 211     Add 36
205     ISZ 212     Skip if Tally is 0
206     JMP 204     Add another 36 if not done
207     HLT         Stop after 18 times
210     0022        Equal to 18(10)
211     0044        Equal to 36(10)
212     0000        Holds the tally
```

Writing the above program was greatly simplified because mnemonic codes were used for the octal instructions. However, writing down the absolute address of each instruction is clearly an inconvenience. If the programmer later adds or deletes instructions, thus altering the location assignments of his program, he has to rewrite those instructions whose operands refer to the altered assignments. If the programmer wishes to move the program to a different section of memory, he must rewrite the program. Since such changes must be made often, especially in large programs, a better means of assigning locations is needed. The assembler provides this better means.

Location Assignment

As in the previous program example, most programs are written in successive memory locations. If the programmer assigned an absolute location to the first instruction, the assembler could be told to assign the next instructions to the following locations in order. In programming the PDP-8/E the initial location is denoted by a precedent asterisk (*). The assembler maintains a current location counter by which it assigns successive locations to instructions. The asterisk causes the current location counter to be set to the value following the asterisk. With this improvement incorporated, the previous example appears as shown in the following example.

```
*200
    CLA CLL
    TAD 210
    CIA
    DCA 212
    TAD 211
    ISZ 212
    JMP 204
    HLT
    0022
    0044
    0000
```

NOTE: In this example, CLA CLL is stored in location 200 and the successive instructions are stored in 201, 202, etc.

WRITING SUBROUTINES

Included in the memory reference instructions, given in Chapter 3, was the instruction JMS (jump to subroutine). This instruction is a modified JMP command which makes return to the point of departure from the main program possible. The JMS instruction automatically stores the location of the next instruction after the JMS in the location to which the program is instructed to jump, thereby enabling a return.

The programmer need only terminate the subroutine with an indirect JMP to the first location of the subroutine in order to return to the next instruction following the JMS instruction. The following simple program illustrates the use of a subroutine to double a number contained in the accumulator.

(Main Program)

```
START,    CLA CLL
          TAD N.          Get the number in the AC
          JMS DOUBLE      Jump to subroutine to double N
          DCA TWON        First instruction after the subroutine
              .
              .
              .
N,        nnnn            Any number, N
TWON;     nnnn            2N will be stored here
```

Subroutine

```
DOUBLE,   0000
          DCA STORE       Save value of N
          TAD STORE       Get N back in the AC
          CLL RAL         Rotate left, multiplying by 2
          SNL             Did overflow occur?
          JMP I DOUBLE
          CLA CLL         If overflow occurs, display the num-
          TAD STORE       ber to be doubled in the AC and
          HLT             then stop the computer.
STORE,    0000
$
```

Notice that the first instruction of the subroutine is located in the second location of the subroutine. Any instruction stored in location DOUBLE would be lost when the return address is stored. Also note that the subroutine as it is written must be located on page 0 or current page, because it is directly addressed. (A subroutine is often located on another page and addressed indirectly as the next example demonstrates.)

The following program multiplies a number in the accumulator by a number stored in the location immediately following the JMS instruction.

Main Program

```
*200
START,          TAD A
                DCA .+3
                TAD B
                JMS I 30
                0000
                DCA PRDUCT
                .
                .
                .
PRDUCT,         0000
A,              0051
B,              0027
*30
                MULT
```

Subroutine

```
*6000
MULT,           0000
                CIA
                DCA MTALLY
                TAD I MULT
                ISZ MTALLY
                JMP .-2
                ISZ MULT
                JMP I MULT
MTALLY,         0000
```

The preceding example illustrates the following important points.

a. The JMS I 30 instruction could be used anywhere in core memory to jump to this subroutine because the pointer word (stored in location 30) is located on page 0, and all pages of memory can reference page 0.
b. The period was used to denote the current location in the instructions DCA .+3 and JMP .—2.
c. Since the result of the subroutine is left in the AC when jumping back to the main program, the next instruction should store the result for future use.
d. The first instruction of the subroutine is in location MULT + 1 since the next address in the main program is stored in MULT by the JMS instruction.
e. The first two instructions of the subroutine set the tally with the negative of the number in the AC.
f. The second number to be multiplied is brought into the subroutine by the TAD I MULT instruction, as it is stored in the location specified by the address that the JMS instruction automatically stores in the first location of the subroutine. This is a common technique for transferring information into a subroutine.

g. The ISZ MTALLY instruction is used in the subroutine to count the number of additions. The ISZ MULT instruction is used to Increment the contents of MULT by one, thereby making the return jump (JMP I MULT) proceed to the next instruction after the location that held the number to be multiplied.

h. An interesting modification of the previous program is achieved through defining a "new operation" MLTPLY by including in the coding the statement MLTPLY = JMS I 30. The assembler would make a replacement in such a way that any time the programmer writes MLTPLY the computer would perform a jump to the subroutine and return to the program with the product in the AC.

ADDRESS MODIFICATION

A very powerful tool often used by the programmer is address modification, meaning the inclusion of instructions in a program to modify the operand portion of a memory reference instruction. It is a particularly useful technique when working with large blocks of stored data as illustrated by the two programs that follow (see Figure 4-8).

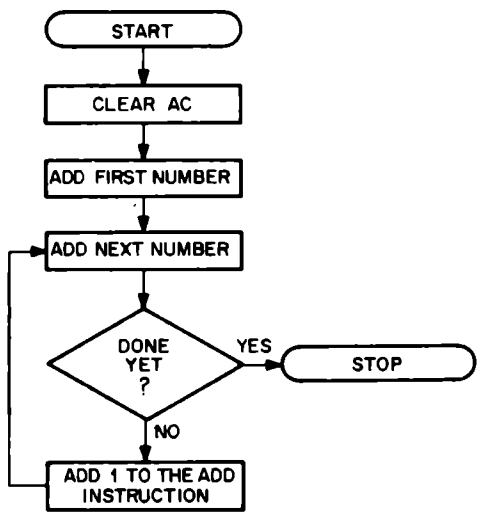


Figure 4-8 Address Modification

```
*200
START,          CLA CLL
                TAD K200
                CIA
                DCA TALLY
                TAD K4000
                DCA NUM
                TAD K4200
                DCA RESULT
```

```
AGAIN,          TAD I NUM
                JMS SQUARE
                DCA I RESULT
                ISZ RESULT
                ISZ NUM
                ISZ TALLY
                JMP AGAIN
                HLT
K200,           0200
TALLY,          0000
K4000,          4000
NUM,            0000
K4200, 4200
RESULT,         0000
*300
SQUARE,         0000
                DCA STORE
                TAD STORE
                CIA
                DCA COUNT
                TAD STORE
                ISZ COUNT
                JMP .-2
                JMP I SQUARE
STORE,          0000
COUNT,          0000
$
```

It will be noted that the first eight instructions are concerned with initializing the program. This initializing enables the stored program to be restarted several times and still operate on the correct locations. If the program had merely incremented locations K4000 and K4200 and utilized those locations for indirect addressing, it would only operate on the correct locations on the first running. On successive runnings, the program would be operating on successively higher locations in memory. With the program written as shown, however, the pointer words are automatically reset. This procedure is often referred to as "housekeeping."

LOOPING A PROGRAM

As many examples have already shown, the use of a program loop, in which a set of instructions is performed repeatedly, is common programming practice. Looping a program is one of the most powerful tools at the programmer's disposal. It enables him to perform similar operations many times using the same instructions, thus saving memory location because he need not store the same instructions many times. Looping also makes a program more flexible because it is relatively easy to change the number of loops required for varying conditions by resetting a counter. It should be remembered that looping is little more than a jump to an earlier part of the program; however, the jump is usually controlled by changing program conditions.

There are two basic methods of creating a program loop. The first me-

thod is using an ISZ (2nnn(octal)) instruction to count the number of passes made through the loop. The ISZ is usually followed by a JMP instruction to the beginning of the loop. This technique is very efficient when the required number of passes through the loop can be readily determined.

The second technique is to use the Group 2 Operate Microinstructions to test conditions other than the number of passes which have been made. Using this second technique, the program is required to loop until a specific condition is present in the accumulator or link bit, rather than until a predetermined number of passes are made.

To illustrate the use of an ISZ instruction in a program loop situation, consider the following program which simply sets the contents of all addresses from 2000 to 2777 to zero.

```
*200
CLEAR,      CLA
            TAD CONST
            DCA COUNT           /SET COUNT TO -1000.
            TAD TTABLE
            DCA STABLE          /SET STABLE TO 2000.
            DCA I STABLE        /CLEAR ONE LOCATION.
            ISZ STABLE          /SELECT NEXT LOCATION.
            ISZ COUNT           /IS OPERATION COMPLETE?
            JMP .-3             /NO: REPEAT.
            HLT                 /YES: HALT.
CONST,      7000                /2'S COMP OF 1000.
COUNT,      0
TTABLE,     TABLE
STABLE,     0                   /POINTER TO TABLE.
*2000
TABLE,      0
$
```

Several points should be carefully noted.

a. The first five instructions initialize the loop, but are not in it. The location COUNT is set to -1000 at the beginning, and 1 is added to it during each passage of the loop. After the 1000th (octal) passage, COUNT goes to zero, and the program skips the JMP instruction, and executes the HLT instruction. On each previous occasion, it executed the JMP instruction.

b. In the list of constants following the HLT instruction, TTABLE contains TABLE, which is defined below has having the value 2000, and containing 0. Therefore, STABLE contains 2000 initially. In order to understand this point, it must be remembered that an asterisk character causes the first location after the asterisk to be set to the value after the asterisk. Therefore, in the previous example CLEAR equals 200 and TABLE equals 2000.

c. ISZ STABLE adds 1 to the contents of location STABLE, forming 2001 on the first pass, 2002 on the second pass, and so on. Since it never reaches zero, it will never skip. This is a very

common use. It is said to be indexing the addresses from 2000 to 2777. (When using an ISZ instruction in this way, the programmer must be certain that it does not reach 0. Follow the ISZ instruction with a NOP if it does reach 0 so that the resulting skip will not modify the program sequence.)

d. For every ISZ instruction used in a program, there must be two initializing instructions before the loop, and there must be a constant and a counting location in a table of constants. This procedure allows the program to be rerun with the counting locations reset to the correct values.

The following program utilizes a Group 2 skip instruction to create a loop. The program will search all of core memory to find the first occurrence of the octal number 1234.

```
*0
NUMBER,     1234
*200
BEGIN,      CLA CLL
            TAD NUMBER
            CIA
            DCA COMPARE         /STORES MINUS NUMBER.
            DCA ENTRY           /SETS ENTRY TO 0.
REPEAT,     ISZ ENTRY           /INCREASES ENTRY.
            CLA
            TAD I ENTRY         /COMPARISON IS
            TAD COMPARE         /DONE HERE.
            SZA CLA
            JMP REPEAT
            TAD ENTRY
            HLT                 /ENTRY IS IN AC.
COMPARE,    0
ENTRY,      0
$
```

This example shows that the program searches itself as well as all other core memory locations, and points up the following points:

a. The ISZ entry instruction is used to index the locations to be tested. The next instruction (CLA) is unnecessary; thus, if ENTRY becomes zero during the course of the program, the program will not be affected. It is important to protect against an ISZ instruction going to zero and skipping a necessary part of a program, if the ISZ is being used simply to index.

b. The number to be searched for is stored in location 0, and the search starts in location 1. Therefore, the program will find at least one occurrence of the number, and will halt after one complete pass through memory, if not before.

c. The program could be modified to bound the area of the search. If the contents of ENTRY are set equal to one less than the desired start location and the number being searched for is put in the location following the last location to be searched, the program will search only the designated area of memory.

d. The program could be restarted at location REPEAT in order to find a second occurrence of 1234 after being halted by the first occurrence.

AUTO-INDEXING

The PDP-8/E computer has eight special registers in page 0; locations 0010 through 0017. Whenever these locations are addressed indirectly by a memory reference instruction, the content of the register is incremented before it is used as the operand of the instruction. These locations can, therefore, be used in place of an ISZ instruction in an indexing application. Because of this, these eight locations are called autoindex registers. Autoindex registers act as any other location when addressed directly. The autoindexing feature is performed only when the location is addressed indirectly.

The following examples below are a modification of the first program example in the preceding section with an autoindex register used in place of the ISZ instruction. (The purpose of the program is to clear memory locations 2000 through 2777.)

Carefully notice the difference between the two examples, especially that TABLE now has to be set to TABLE-1 since this is incremented by the autoindexing register before being used for the first time. This point must be remembered when using an autoindex register. The register increments before the operation takes place; therefore, it must always be set to one less than the first value of the addresses to be indexed.

```
*10
INDEX,      0
*200
CLEAR,      CLA
            TAD CONST
            DCA COUNT
            TAD TTABLE
            DCA INDEX
            DCA I INDEX
            ISZ COUNT
            JMP .-2
            HLT
CONST,      7000
COUNT,      0
TTABLE,     TABLE-1
*2000
TABLE,      0
$
```

The memory search example of the preceding section could also be simplified using an autoindex register as shown below.

```
*0
NUMBER,     1234
*10
ENTRY,      0
*200
BEGIN,      CLA CLL
            TAD NUMBER
```

Notice that in this case ENTRY originally equals 0 because its content is incremented before being used to obtain data for the comparison.

```
            CIA
            DCA COMPARE
            DCA ENTRY
REPEAT,     TAD I ENTRY
            TAD COMPARE
            SZA CLA
            JMP REPEAT
            TAD ENTRY
            HLT
COMPARE,    0
$
```

PROGRAM DELAYS

Because computer development has been primarily sparked by a desire for speed in performing calculations, it seems inconsistent and self-defeating to slow the computer down with program delays. However, there are many occasions when a computer must be told to slow down or to wait for further information. This is because most peripheral equipment, and certainly the human operator, is very much slower than the computer program. A temporary delay may be introduced into the execution of a program when needed by causing the computer to enter one or more futile loops, which it must traverse a fixed number of times before jumping out. It is often necessary to have a computer perform a temporary delay while a peripheral device is processing data to be submitted to the computer. The delays can be accurately timed so as not to waste any more computer time than necessary.

The following is a simple delay routine using the ISZ instruction for an inner loop and an outer loop. When analyzing the example it should be remembered that the PDP-8/E represents only positive numbers up to 3777(octal) or 2047(10). Therefore, the computer counts up to 2047 (10) and then continues to count starting at the next octal number 4000(octal), which the computer interprets as —2048(10). Successive increments of this number will finally bring the count to zero. Thus, a location could be used to count from 1 up to 0 by using an ISZ instruction.

```
                (main program)
                    .
                    .
                    .
            TAD CONST       /START OF DELAY ROUTINE
            DCA COUNT
            ISZ COUNT 1     /INNER
            JMP .—1         /LOOP
            ISZ COUNT
            JMP .—3
CONST,      6030            /SETS DELAY
COUNT,      0
COUNT 1     0
```

PROGRAM BRANCHING

Very few meaningful programs are written which do not take advantage

of the computer's ability to determine the future course the program should follow, based upon intermediate results. The procedure of testing a condition and providing alternative paths for the program to travel for each of the different results possible is called branching a program. The Group 2 microinstructions presented previously are most often used for this purpose. The ISZ instruction often referred to as a conditional skip instruction, also provides a branch in a program. This instruction operates upon the contents of a memory location, while the Group 2 microinstructions test the contents of the AC and L.

A typical example of a conditional skip would be a program to compare A and B and to reverse their order if B is larger than A (see Figure 4-9).

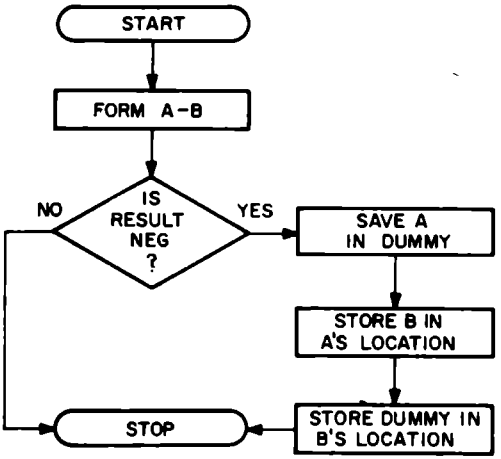


Figure 4-9 Conditional Skip

```
*200
TEST,     CLA CLL
          TAD B          /SUBTRACT B
          CIA            /FROM A
          TAD A          /HERE.
          SMA CLA
          HLT            /STOP HERE IF A IS GREATER
                         /OR EQUAL
          TAD A          /THE REMAINDER OF
          DCA DUMMY      /THE PROGRAM
          TAD B          /DOES THE SWITCH.
          DCA A
          TAD DUMMY
          DCA B
          HLT
A,        1234           /SUBSTITUTE ANY POSITIVE
B,        2460           /VALUES FOR A AND B.
DUMMY,    0
$
```

If A is less than B, their difference will be negative and the HALT will be skipped. The program will proceed to reverse the order of A and B. If A is greater than or equal to B, the program will halt.

MICROPROGRAMMING

Because PDP-8/E instructions of Group 1, Group 2, and Group 3 are determined by bit assignment, these instructions may be combined, or microprogrammed, to form new instructions enabling the computer to do more operations in less time.

Combining Microinstructions

The programmer should make certain that the program clears the accumulator and link before any arithmetic operations are performed. To perform this task, the program might include the following instructions (given in both octal and mnemonic form).

```
CLA        7200 (octal)
CLL        7100 (octal)
```

However, when the Group 1 instruction format is analyzed, Figure 4-10 illustrates the result.

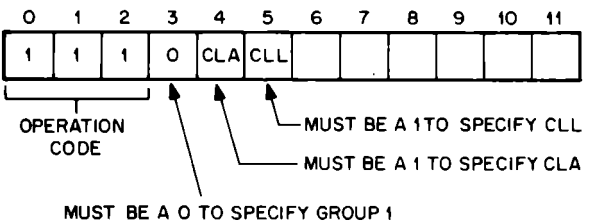

0 1 2 3 4 5 6 7 8 9 10 11
1 1 1 0 CLA CLL
OPERATION CODE
MUST BE A 1 TO SPECIFY CLL
MUST BE A 1 TO SPECIFY CLA
MUST BE A 0 TO SPECIFY GROUP 1

Figure 4-10 Group 1 Bit Assignments

Since the CLA and the CLL instructions occupy separate bit positions, they may be expressed in the same instruction, thus combining the two operations into one instruction. This instruction would be written as follows.

```
CLA CLL    7300 (octal)
```

In this manner, many operate microinstructions can be combined, making the execution of the program much more efficient. The assembler for the PDP-8/E will combine the instructions properly when they are written as above; that is, on the same coding line, separated by a space.

Illegal Combinations

Microprogramming, although very efficient, can also be troublesome for the new programmer. There are many violations of coding which the assembler will not accept.

One rule to remember is: "If you can't code it, the computer can't do it." In other words, the programmer could write a string of mnemonic microinstructions, but unless these microinstructions can be coded

correctly in octal representation, they cannot be performed. To illustrate this fact, suppose the programmer would like to complement the accumulator (CMA), complement the link (CML), and then skip on a non-zero link (SNL). He could write the following.

```
CMA     CML     SNL
```

These instructions require the bit assignments shown in Figure 4-11.

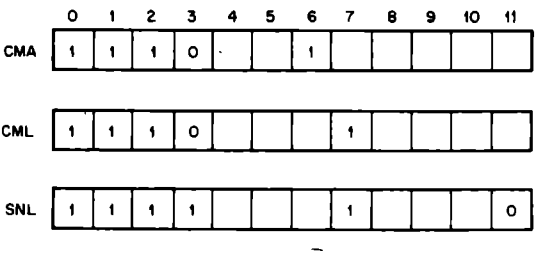


Figure 4-11 Example of Illegal Combinations

The three microinstructions cannot be combined in one instruction because bit 3 is required to be a 0 and a 1 simultaneously. Therefore, no instructions may be used which combine Group 1 and Group 2 microinstructions because bit 3 usage is not compatible. The CMA and CML can, however, be combined because their bit assignments are compatible. The combination would be as follows.

```
CMA  CML      7060 (octal)
```

To perform the original set of three operations, two instructions are needed.

```
CMA CML       7060 (octal)
SNL           7420 (octal)
```

Because Group 1 and Group 2 microinstructions cannot be combined, the commonly used microinstruction CLA is a member of both groups. Clearing the AC is often required in a program and it is very convenient to be able to microprogram the CLA with the members of both groups.

The problem of bit assignment also arises when some instructions within a group are combined. For example, in Group 1 the rotate instructions specify the number of places to be rotated by the state of bit 10. If bit 10 is a 0, rotate one place; if bit 10 is a 1, rotate two places. Thus, the instruction RAR can not be combined with RTR because bit 10 would be required to have two different values at once. If the programmer wishes to rotate right three places, he must use two separate instructions.

```
RAR           7010 (octal)
RTR           7012 (octal)
```

Although he can write the instruction "RAR RTR," it cannot be correctly converted to octal by the assembler because of the conflict in bit 10; therefore, it is illegal.

Combining Skip Microinstructions
Group 2 operate microinstructions use bit 8 to determine the instruction specified by bits 5, 6, and 7 as previously described. If bit 8 is a 0, the instructions SMA, SZA, and SNL are specified. If bit 8 is a 1, the instructions SPA, SNA, and SZL are specified. Thus, SMA cannot be combined with SZL because of the opposite values of bit 8.

OR GROUP—SMA OR SZA OR SNL
If bit 8 is a 0, the instruction skips on the logical OR of the conditions specified by the separate microinstructions. The next instruction is skipped if any of the stated conditions exist. For example, the combined microinstruction SMA SNL will skip under the following conditions:

a. The accumulator is negative, the link is zero.
b. The link is nonzero, the accumulator is not negative.
c. The accumulator is negative and the link is nonzero.

(It will not skip if all conditions fail.) This manner of combining the test conditions is described as the logical OR of the conditions.

AND GROUP—SPA AND SNA AND SZL
A value of bit 8 = 1 specifies the group of microinstructions SPA, SNA, and SZL, which combine to form instructions that act according to the logical AND of the conditions. In other words, the next instruction is skipped only if all conditions are satisfied. For example, the instruction SPA SZL will cause a skip of the next instruction only if the accumulator is positive and the link is zero. (It will not skip if either of the conditions fail.)

NOTES: 1. The programmer is not able to specify the manner of combination. The SMA, SZA, SNL conditions are always combined by the logical OR, and the SPA, SNA, SZL conditions are always joined by a logical AND.
2. Since the SPA microinstruction will skip on either a positive or a zero accumulator, to skip on a strictly positive (positive, nonzero) accumulator the combined microinstruction SPA SNA is used.

Order of Execution of Combined Microinstructions
The combined microinstructions are performed by the computer in a very definite sequence. When written separately, the order of execution of the instructions is the order in which they are encountered in the program. In writing a combined instruction of Group 1 or Group 2 microinstructions, the order written has no bearing upon the order of execution. This should be clear, because the combined instruction is a 12-bit binary number with certain bits set to a value of 1. The order in which the bits are set to 1 has no bearing on the final execution of the whole binary word.

GROUP 1

1. CLA, CLL—Clear the accumulator and/or clear the link are the first actions performed. They are effectively performed simultaneously and yet independently.
2. CMA, CML—Complement the accumulator and/or complement the link. These operations are also effectively performed simultaneously and independently.
3. IAC—Increment the accumulator. This operation is performed third, allowing a number in the AC to be complemented and then incremented by 1, thereby forming the two's complement, or negative, of the number.
4. RAR, RAL, RTR, RTL, BSW—The rotate instructions are performed last in sequence. Because of the bit assignment previously discussed, only one of the five operations may be performed in each combined instruction.

GROUP 2

1. Either SMA or SZA or SNL when bit 8 is a 0. Both SPA and SNA and SZL when bit 8 is a 1. Combined microinstructions specifying a skip are performed first. The microinstructions are combined to form one specific test; therefore, skip instructions are effectively performed simultaneously.
   Because of bit 8, only members of one skip group may be combined in an instruction.
2. CLA—Clear the accumulator. This instruction is performed second in sequence, allowing different arithmetic operations to be performed after testing (see Event 1) without the necessity of clearing the accumulator with a separate instruction before some subsequent arithmetic operation.
3. OSR—Inclusive OR between the switch register and the AC. This instruction is performed third in sequence, allowing the AC to be cleared first, and then loaded from the switch register.
4. HLT—The HLT is performed last to allow any other operations to be concluded before the program stops.

This is the order in which all combined instructions are performed. In order to perform operations in a different order, the instructions must be written separately, as shown in the following example. The following combined microinstruction looks as if it might clear the accumulator, perform an inclusive OR between the SR and the AC, and then skip on a nonzero accumulator.

```
CLA OSR SNA
```

However, the instruction would not perform in that manner, because the SNA would be executed first. In order to perform the skip last, the instructions must be separated, as follows:

```
CLA OSR
SNA
```

Microprogramming requires that the programmer carefully code mnemonics legally so that the instruction actually does what he desires it

to do. The sequence in which the operations are performed and the legality of combinations are crucial to PDP-8/E programming.

The following is a list of commonly used combined microinstructions, some of which have been assigned a separate mnemonic.

	INSTRUCTION	EXPLANATION
—	CLA CLL	Clear the accumulator and link.
CIA	CMA IAC	Complement and increment the accumulator. (Sets the accumulator equal to its own negative.)
LAS	CLA OSR	Load accumulator from switches. (Loads the accumulator with the value of the switch register.)
STL	CLL CML	Set the link (to a 1).
—	CLA IAC	Sets the accumulator to a 1.
STA	CLA CMA	Sets the accumulator to a −1.

In summary, the basic rules for combining operate microinstructions are as follows:

a. Group 1 and Group 2 microinstructions cannot be combined.
b. Rotate microinstructions (Group 1) cannot be combined with each other.
c. OR Group (SMA, SZA, or SNL) microinstructions cannot be combined with AND Group (SPA, SNA, or SZL) microinstructions.
d. OR Group microinstructions are combined as the logical OR of their respective skip conditions. AND Group microinstructions are combined as the logical AND of their respective skip conditions.
e. Order of execution for combined instructions is listed below.

Group 1	Group 2
1. CLA, CLL	1. SMA/SZA/SNL (OR group) or SPA/SNA/SZL (AND group)
2. CMA, CML	2. CLA
3. IAC	3. OSR
4. RAR, RAL, RTR, RTL, BSW	4. HLT

SYSTEM PROGRAMMING

The Programming System for the PDP-8/E consists of SYSTEM PROGRAMS, UTILITY PROGRAMS, and APPLICATION PROGRAMS, and is complemented with the DECUS LIBRARY (see Figure 4-12).

More than 1000 PDP-8 programs are available to the user. Digital Equipment Corporation's Program Library, for instance, offers more than 700 programs from which to choose. In addition, a library containing programs developed by PDP-8 users, called the "DECUS LIBRARY," is available to all PDP-8 users. These programs cover a wide variety of applications in addition to the application programs developed by DEC.

The programming system was designed to simplify and accelerate the process of learning to program. At the same time, experienced programmers will find that it incorporates many advanced features. The system is intended to make immediately available to each user the full,

general-purpose data processing capability of the computer and to serve as the operating nucleus for a growing library of programs and routines available to all installations.

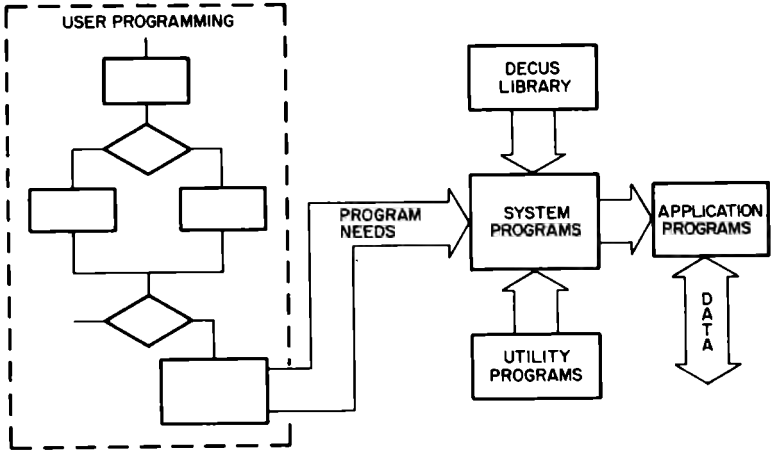


Figure 4-12 PDP-8/E Programming System

PDP-8/E Software Kit
A basic Family-of-8 software kit for the PDP-8/E computer is provided with each basic PDP-8/E system. The user receives one or more paper tapes and the necessary documentation for each program. Those programs provided in the basic software kit are identified by an asterisk (*) in the following program descriptions.

System Programs
The system programs are used, combined with the utility programs and/or the DECUS Library Program, to translate the user's ideas into desired application programs. System Programs include: Monitors, Editors, Assemblers, Compilers, Interpretive Languages, Debuggers, and Loaders.

Monitor Programs
Monitor programs used with the PDP-8/E include PS/8, Disk Monitor, and TSE.

PS/8 PROGRAMMING SYSTEM—PS/8, an 8K programming system, represents a significant advance in software development for small computers with capabilities which were formerly available only on such powerful machines as the PDP-10.

The PS/8 is a program development system for the PDP-8/E with minimum 8K of core and one or more of the following mass storage devices:

a. TC08/TU56 DECtape
b. RF08 Disk
c. RK8 Disk Pack

The 8K programming system has the following features:

a. It allows the user device-independent access to up to 15 I/O devices, including up to eight DECtapes, up to four Disk units, Teletype, high speed paper tape reader and punch, card reader, line printer, and any other device for which it is possible to write a device handler in one, or in some cases two, pages.
b. The user program may call upon the monitor for several services, including loading device handlers for devices to which the user may assign a name, looking up input files on these devices, creating and closing variable and fixed-length output files on these devices, and getting and decoding a line of input from the console Teletype that identifies input and output files and options.

PS/8 gives the user, in addition to the language processor (8K FOCAL, 8K FORTRAN, 8K PAL-D, and SABR), absolute and relocatable loaders, a symbolic editor, CONVERT (a program to provide file compatibility with the present Disk Monitor System), PIP (Peripheral Interchange Program), and an invisible ODT (Octal Debugging Technique), which allows the programmer to debug programs without giving up valuable core space.

Advantages offered by the PS/8 System include:

a. Device Independence—
    1. Programs can be run using the most effective I/O devices available at a given installation.
    2. As an installation grows, programs need not be rewritten for increased I/O capability.
    3. A user device handler may easily be added.
b. User Interfacing—
    1. User may use any standard I/O device without having to directly program the device.
    2. User may use the system command decoder to vary the I/O used in the programs.
c. Performance—Increased, due to efficient use of storage devices (especially noticeable on DECtape).
d. Expandability—As the system grows, so does the capability of the PS/8 system.
e. The PS/8 Library of Programs.
f. Increased power, derived from the addition of 4K of core to an existing 4K disk monitor system.
g. A program to convert all 4K monitor files to PS/8 files.
h. The capability of accommodating any amount of core storage from 8K to 32K.

Disk Monitor System
A keyboard monitor is available to Disk System users that allows them to save core images on the Disk and restore them to memory. The extensive software package available consists of: FORTRAN Compiler, Program Assembly Language (PAL-D), Editor Program (Editor), Peripheral Interchange Program (PIP), and Dynamic Debugging Technique (DDT-D) Program.

In addition, the user may save and restore his own core images and use the remainder of the available device storage for temporary storage of source or binary data. The monitor system may also be used with DECtape.

Time-Sharing Monitor

TSE (Time-Sharing PDP-8/E) is a general purpose, stand-alone monitor time-sharing system. The TSE can accommodate up to 16 users simultaneously. A minimum of 12K of core memory and an RF08-type disk is required for a comprehensive library of system programs, which provide facilities for compiling, assembling, editing, loading, saving, calling, debugging, and running user programs on line.

The center of a TSE system is a complex of programs called Monitor. Monitor coordinates the operations of the various units, allocates the time and services of the computer to users, and controls their access to the system. The computer works on user programs simultaneously by segregating the central processor operations from the time-consuming interactions of the human users. Execution of various programs are interspersed without interfering with one another and without detectable delays in the responses to the individual user.

Consult the DECUS Library for additional monitor programs.

Editor Program

Symbolic Paper Tape Editor*

The Symbolic Paper Tape Editor program is used to edit, correct, and update symbolic program tapes using the PDP-8/E, the Teletype unit, and/or the high-speed reader. With Editor in core memory, the user reads in portions of his symbolic tape, removes, changes, or adds instructions, and gets back a complete new symbolic tape with errors removed. He can work through the program instruction, spot check it, or concentrate on new sections. A character string search is available. The user can move one or more lines of text from one place to another.

The program requires 4K core and a Teletype. Consult the DECUS Library for additional editor programs.

*Part of the basic software package.

Assembler Programs

Assembler Programs used with the PDP-8/E include PAL-III, PAL-D, MACRO-8, and 8K-SABR. The use of an assembly program has become a standard practice in programming digital computers. This process allows the programmer to code instructions in a symbolic language, one he can work with more conveniently then the 12-bit binary numbers that actually operate the computer. The assembly program translates the symbolic language program into its machine code equivalent. The advantages are significant: the symbolic language is more meaningful and convenient to a programmer than a numeric code; instructions or data can be referred to by symbolic names without concern for, or even knowledge of, their actual addresses in core memory; decimal and alphabetical data can be expressed in a form more convenient than binary numbers; programs can be altered more efficiently and debugging is considerably simplified.

PAL-III*—PAL-III is a basic assembler allowing symbolic references, origins, and expressions. The output is in a form suitable for input to the binary loader.

PAL-III is a two-pass assembler with an optional third pass; i.e., the symbolic program tape must be passed through the assembler two times to produce the binary-coded tape, and the optional third pass produces a complete octal symbolic program listing, which can be typed and/or punched.

PAL-III accepts symbolic program tapes from either the low-speed or high-speed reader and produces the binary tapes on either the low-speed or high-speed punch.

During assembly, the programmer communicates with PAL-III via the switches on the computer console. Switch options are used to specify which pass the assembler is to perform and which reader and punch the assembler should accept input from and punch out on.

PAL-III requires 4K of core memory and a Teletype.

PAL-D—PAL-D incorporates most of the features of both PAL-III and MACRO-8, and is used only in the Disk Monitor System. PAL-D is designed primarily for 4K PDP-8/E computers with disk or DECtape.

An 8K version of PAL-D is available for use with either 8K or 12K of core memory.

MACRO-8*—MACRO-8 is an advanced assembler which has the same features as PAL-III, plus the following additional features: user-defined macros, double precision integers, floating-point constants, arithmetic and Boolean operators, literals, text facilities, and automatic off-page linkage generation. To incorporate such features, the size of the user's symbol table was decreased. However, the programmer can increase or decrease the size of the permanent symbol table at the expense of some of the more space-consuming features.

MACRO-8 requires 4K of core memory and a Teletype.

*Part of the basic software package.

8K SABR—8K SABR (Symbolic Assembler for Binary Relocatable Programs) is an advanced one-pass symbolic assembler. It translates symbolic programs written in the SABR language into binary relocatable code acceptable to the computer. SABR programs are core page independent. Therefore, programs may be written without regard to the 128-word core page of the computer. SABR automatically generates off-page and off-field references for direct or indirect statements. It also automatically connects instructions on one page to those that overflow onto the next. The list of available pseudo-ops is extensive, including external subroutine calling, argument passing, and conditional assembly. SABR offers an optional second pass to produce a side-by-side octal/symbolic listing of the assembled program.

The relocatable binary tapes produced by SABR are loaded into any field of core memory using the 8K linking Loader, as are the compre-

hensive library of subprograms. These subprograms may be called by any SABR program.

The high speed reader and punch is strongly recommended for the 8K SABR program.

Consult the DECUS Library for additional assembler programs.

Compiler Programs

Compiler programs used with the PDP-8/E include DIBOL, FORTRAN 4K and 8K, and ALGOL-8. Although the DIBOL program is listed with compiler programs, DIBOL includes a package which contains a Monitor Program, an Editor Program, and utility programs.

DIBOL Software System—DIBOL (Digital Equipment Business-Oriented Language) is a complete business-oriented software system for the PDP-8/E. It allows a relatively inexperienced programmer to produce complete business application programs on a small computer. The DIBOL software system is directed toward the small business which can write its own in-house EDP management/accounting programs. The small business can now use the power of the PDP-8/E computer to efficiently and economically construct programs for billing, accounts receivable, inventory control, payroll, general ledger, etc.

The DIBOL software system contains a simple business-oriented language, a data management system to provide automatic input, sorting, and file maintenance, and a monitor to tie the various subsystems together.

DIBOL is a simple, powerful language which provides the means to an end for such operations as accounting and management reporting: it is not an end in itself. That is, the DIBOL software system is a useful data processing package built around the DIBOL language in the same way that COBOL is used on other machines.

DIBOL combines the best features of COBOL and FORTRAN into an easy-to-learn business language. COBOL is organized into two principal sections; a set up file section and a data manipulation section. DIBOL follows this same organization. DIBOL, like COBOL, can perform all necessary file/data manipulation commands, convert alphanumeric to decimal statements, and convert decimal to alphanumeric statements, GO-TO statements, and subroutine calls. However, DIBOL commands are in simple algebraic expressions similar to FORTRAN, whereas COBOL commands are written out completely.

The exciting concept of DIBOL is that it is easy to learn and requires a minimum amount of training.

Minimum Machine for Program Preparation —

PDP-8/E
8K Core
TC08 DECtape Control
2 TU56 DECtapes
ASR-33 Teletypes

Input Terminal—Paper Tape Reader or Card Reader

Output Terminal—Line Printer (If no line printer, then an ASR-35 is recommended)

A Disk is not accommodated by this system. 8K core only is utilized in this version.

FORTRAN Compiler (4K)*—The 4K FORTRAN (for FORmula TRANslation) compiler lets the user express the problem he is trying to solve in a mixture of English words and mathematical statements that is close to the language of mathematics and is also intelligible to the computer. In addition to reducing the time needed for program preparation the compiler enables users with little or no knowledge of the computer's organization and operating language to write effective programs for it. The FORTRAN Compiler contains the instructions the computer requires to perform the clerical work of translating the FORTRAN version of the program statements into machine language object programs. It also produces diagnostic messages. After compilation, the object program, the operating system, and the data with which it works are loaded into the computer for solution of the problem.

The program requires 4K of core memory and a Teletype.

*Part of the basic software package.

FORTRAN Compiler (8K)—The 8K FORTRAN Compiler is an extension of the 4K FORTRAN Compiler which features the following additions:

a) U.S.A. Standard FORTRAN Syntax
b) Subroutines
c) Two levels of subscripting
d) Function subprograms
e) I/O Supervisor
f) Relocatable Linking Loader Output
g) Common I,E,F,H,A,X format specification
h) Arithmetic and trigonometric library

This compiler utilizes all available core from 8K to 32K, and correctly loads programs over field boundaries.

The program requires 8K of core memory and a Teletype.

ALGOL-8 (available through DECUS only)—ALGOL (ALGOrithmic Language) is one of the most widely used international programming languages. The language emphasizes formal, well-defined procedures for solving problems with computers, and is the standard language of the Association for Computing Machinery (ACM). ALGOL-8 Compiler generally conforms to SUBSET ALGOL 60 as approved by the International Federation of Information Processing Societies (IFIPS).

ALGOL-8 requires 4K of core memory and a Teletype.

Consult the DECUS Library for additional compiler programs.

Interpretive Programs—Interpretive Programs used with the PDP-8/E include FOCAL and BASIC.

FOCAL-8*—FOCAL-8 (FOrmula CALculator) is an on-line conversational language designed for solving complex calculations. The language consists of short imperative English statements and mathematical expressions in standard notation. FOCAL puts the full calculating power and speed of the computer at the user's fingertips without the user having to master the intricacies of machine-language programming. FOCAL is an easy way of simulating mathematical models, plotting curves, handling set of simultaneous equations in N-dimensional arrays, and much more.

FOCAL is available in several configurations.

a) Single-user FOCAL—requires 4K and Teletype.
b) Four-user FOCAL—requires 8K core and four Teletypes (console plus three).
c) Seven-user FOCAL—requires 8K core, seven Teletypes (console plus six), and a RF08 or DF32D disk.

*Part of the basic software package.

BASIC—BASIC is available in two versions:

a) BASIC-8 is only available with TSE
b) BASIC is available through DECUS and requires 4K and Teletype, and either DECtape or Disk (RF08 or DF32-D).

BASIC-8 is a modified version of the algebraic language developed at Dartmouth College. The BASIC language is composed of easy-to-learn English statements and mathematical expressions.

Under development and to be available soon are two additional versions:

a) Batch BASIC: requires a card reader (CR8-E or CM8-E), line printer LE8, 4K core, and either DECtape or Disk (RF08 or DF32-D).
b) Four-user BASIC: requires 8K core, four Teletypes (console plus three), and either DECtape or Disk (RF08 or DF32-D).

Consult the DECUS Library for additional interpretive programs.

Debugger Programs—Debugger Programs used with the PDP-8/E include ODT-8 and DDT-8.

DDT-8 (Dynamic Debugging Techniques)*—On-line debugging with DDT-8 gives the user dynamic printed program status information. It gives him close control over program execution, preventing errors ("bugs") from destroying other portions of his program. He can monitor the execution of single instructions or subsections, change instructions or data in any format, and output a corrected program at the end of the debugging session.

Using the standard Teletype, the user can communicate conveniently with the PDP-8/E in the symbols of his source language. He can control the execution of any portion of his object program by inserting breaks, or traps, in it. When the computer reaches a break, it transfers control of the object program to DDT. The user can then examine and modify

the content of individual core memory registers to correct and improve his object program.

DDT-8 requires 4K of core memory and a Teletype.

*Part of the basic software package.

ODT-8 (Octal Debugging Technique)*—ODT-8 allows the programmer to do all the things mentioned in DDT-8 by communicating with his object program using the octal representation of his binary program. ODT-8 occupies less core storage than DDT-8 and can be loaded in upper memory or lower memory, depending on where the binary program resides.

ODT-8 requires 4K of core memory and a Teletype.

Consult the DECUS Library for additional debugging programs.

LOADERS

Read-In Mode (RIM) Loader*

The RIM Loader is a minimum routine for reading and storing information contained in read-in mode coded tapes via the Teletype or high speed paper tape reader.

Binary Loader*

The Binary Loader is a short routine for reading and storing information contained in binary-coded tapes, using the Teletype or high-speed paper tape reader.

The Binary Loader accepts tapes prepared by the use of PAL or MACRO-8. Diagnostic messages may be included on tapes produced when using either PAL or MACRO. The Binary Loader ignores all diagnostic messages.

See Appendix A and the DECUS Library listing for additional loader programs.

*Part of the basic software package.

Utility Programs

PDP-8/E utility programs provide printouts or punchouts of core memory content in octal, decimal, or binary form, as specified by the user. Subroutines are provided for octal or decimal data transfer and binary-to-decimal, decimal-to-binary, and Teletype tape conversion.

Most of these programs require only 4K of core memory and a Teletype. Some of the primary utility programs include data conversion programs and maintenance and diagnostic programs (see Appendix A for a listing of other utility programs).

Data Conversion Programs—Data conversion programs used with the PDP-8/E include the Floating Point Package and Math Function Routines.

Mathematical Function Routines

The programming system includes a set of mathematical function routines to perform the following operations: single precision multiplication,

division, square root; double precision sine, cosine, multiply, divide; and arithmetic and logical shifts.

Floating Point Package
The Floating Point Package permits the PDP-8/E to perform arithmetic operations that many other computers can perform only after the addition of costly optional hardware. Floating point operands retain the maximum precision available by discarding leading zeros. In addition to increasing accuracy, floating point operations relieve the programmer of scaling problems common in fixed-point operations, a particularly advantageous feature to the inexperienced programmer. The floating point subroutines and interpreter permit the programmer to encode arithmetic operations to either six or ten decimal digits of precision as easily as though the machine had floating point hardware. Also included in the package are input and output conversion routines.

* Part of the basic software package.

MAINTENANCE AND DIAGNOSTIC PROGRAMS
A complete set of standard diagnostic programs is provided (see Appendix A) to simplify and expedite system maintenance. Program descriptions and manuals permit the user to effectively test the operation of the computer for proper core memory functioning and proper execution of instructions. In addition, diagnostic programs to check the performance of standard and optional peripheral devices are provided with the devices.

Consult the DECUS Library for additional utility programs.

Application Programs
Integrated Application Packs are special-purpose software packages which provide a specific solution to one aspect of a general problem. By using the proper application pack and additional hardware and/or software, the user can have a computer system customized to fit his particular requirements. Although numerous programs are offered by the DECUS library, some of the commonly used application programs include:

Display 8
INDAC 8
Communications 8
LAB 8
GLC 8
Educational FOCAL
AP2
Quick Point 8
Typeset 8
PHA 8

The DECUS Library
Although each PDP-8/E is delivered to the user complete with an extensive program kit, the DECUS Library provides the user with a continually growing assortment of various programs which are available for general use. DECUS includes a wide variety of system programs, utility programs, and application programs. A listing of the programs available from the DECUS Library can be obtained from the Program Library, Digital Equipment Corporation, Maynard, Mass. 01754.

CHAPTER 5

PROGRAMMED DATA TRANSFERS

GENERAL

The nature of the IOT instruction was explained in Chapter 2, programming the teletype was defined in Chapter 4, and the discussion of data transfers as viewed from the peripheral will be discussed in Chapters 9 and 10. This chapter deals with programmed data transfers as viewed from the processor. This discussion is directed to the major considerations of setting up the mechanism for transferring data to and from the processor.

Three types of data transfers are offered by the PDP-8/E processor to receive, store, and transmit data between one or more peripherals and the processor. These transfers are called:

a) Programmed I/O Transfers,
b) Programmed Transfers using the Program Interrupt Facility,
c) Data Transfers using the Data Break Facility.

The first two types of transfers, described in this chapter, are controlled by the program, while the data break transfers are controlled by each peripheral. Data break transfers are discussed in Chapter 6.

PROGRAMMED DATA TRANSFER vs. DATA BREAK

Most input/output (I/O) transfers are controlled by the computer program. Such an information transfer requires perhaps five times as much computer time as does a data break transfer. However, in terms of real time, the duration of a programmed transfer is rather small, due to the high speed of the computer, and is usually well within the limits required for laboratory or process control instrumentation. To get maximum benefit from the control features of the PDP-8/E, the user should utilize programmed data transfers in most cases. Moreover, the peripheral control circuits are usually simpler and less expensive than are those of data break transfer peripherals.

PERIPHERAL REQUIREMENTS

Programmed data transfers use the processor accumulator (AC) register as an intermediate storage point between core memory and a buffer register within the peripheral. If an output data transfer is to be performed, the computer program causes data to be loaded from core memory into the AC. The program then uses an input/output transfer (IOT) instruction to, first, select the desired peripheral and, second, direct the peripheral to generate certain control signals, which cause the data in the AC to be placed onto the OMNIBUS DATA lines. The peripheral, whose control module monitors these DATA lines, then strobes the data into its input buffer register and prepares to process the information. Each transfer from computer to peripheral is handled in this manner.

If the transfer is from peripheral to computer, the process is reversed.

The IOT instruction selects the peripheral and directs it to generate control signals. The peripheral then places the data in its output buffer register on the OMNIBUS DATA lines. A processor timing signal strobes the information into the AC. The program may then either deposit the data into a memory location or use it in some other way.

Each peripheral connected to the OMNIBUS transfers data in the manner described. The bus system of I/O transfers, by which many peripherals monitor the OMNIBUS signal lines, imposes the following requirements on the peripheral equipment:

a. Each peripheral must contain a device selector circuit which monitors Memory Data (MD) lines 3 through 8 of the OMNIBUS. During an IOT instruction, these MD lines carry a selection code which is unique for each peripheral and which must be decoded by the peripheral's device selector.
b. Each peripheral must contain gating circuits which monitor the MD9-11 lines of the OMNIBUS. During an IOT instruction, these MD lines carry command signals that the peripheral must translate into transfer control signals.
c. Each peripheral must contain gating circuits at the input of a receiving register and at the output of a transmitting register (the functions of these registers may be realized with a single buffer register if desired). These gating circuits must be capable of strobing data into or out of the registers when triggered by a command from the device selector.
d. Each peripheral must contain a "Busy/Done" flag (a flip-flop) and gating circuit, which together can assert the OMNIBUS SKIP line when commanded by an IOT instruction. When the flag is set, the peripheral is ready for a word transfer.

PRINCIPLES OF PROGRAMMED I/O TRANSFERS

The simplest and most straightforward type of transfer is the programmed transfer. Rather than responding to an interrupt request and checking each flag to find out which device made the request, the program remains in a continuous wait loop. This method is convenient when the purpose of the processor is to service one or more peripherals. If the user desires to do processing in addition to servicing peripherals, he should employ the interrupt facility.

The IOT Instruction

The nature of the IOT instruction is summarized as follows:

a. PROCESSOR OPERATION CODE—IOTs use operation code 6 and are one-cycle augmented instructions.
b. DEVICE SELECTION—The middle six bits of the instruction word are used for device selection; each peripheral decodes these bits and responds to the IOT only if it sees its particular number or device code. (There is an obvious corollary to this statement. Peripherals generally do not share the same device code.)
c. DEVICE OPERATION CODE—The last three bits of the IOT are used to tell the device what to do. The method varies, depending on whether the device plugs into the OMNIBUS or is at-

tached to the External I/O Bus interface. The end result is the same, however, in that the last three bits define the operation to be performed. The usage is as follows:

Binary Value			Octal Value	Conventional Usage
Bit 9	Bit 10	Bit 11 (LSB)		
0	0	1	1	Sampling flags, skipping
0	1	0	2	Clearing flags, clearing AC
1	0	0	4	Reading, loading, and clearing buffers

Flags

In the control section of every I/O device is a flag, or status flip-flop. The flag is cleared when the computer is first turned on. It can be cleared by one of the device's IOTs, and is set whenever the device finishes its operation. The state of this flag can also be tested, usually by an IOT that causes the next instruction to be skipped if the flag is set.

Input devices (from peripheral to processor) set flags when they have data to be serviced by the processor.

Output devices (from processor to peripheral) receive an IOT instruction that clears the flag. When the device has processed the data received from the processor, it then sets the flag (to let the processor know that it is ready for a new instruction).

In order to understand the purpose of a flag, consider the following situation:

A program requires that a number be entered into the program via the Teletype keyboard. Obviously, the operator might elect to take a lot of time deciding what number to enter. Equally obviously, the program must wait until that number has been entered; otherwise, wrong computation will take place. Thus, a synchronization problem exists. The processor must wait until the operator has made up his mind.

This problem is easily overcome by making use of the flag. The flag sets when the keyboard electronics has a character available in its buffer. Furthermore, the program can find out when the flag sets by means of the skip IOT. All that has to be done is to use the following instructions:

```
Sometimes Called a "Wait Loop"   KSF      /SKIP IF THE KEYBOARD FLAG IS SET.
                                 JMP .-1  /IF YOU DIDN'T SKIP THE LAST TIME, TRY AGAIN.
                                          /EVENTUALLY THE PROGRAM WILL GET PAST THE JUMP INSTRUCTION.
                                          /WHEN THE OPERATOR STRIKES A KEY.
```

Note that "JMP .—1" means "jump back to previous instruction." Assuming that the flag becomes set, the program then clears the flag and brings the character into the AC. If it does not clear the flag at this time, the device will not get a second character. Also, the program must

move the character into the accumulator so that it can process the character (store or operate upon it). This is illustrated by:

```
.
KCC          /CLEAR THE KEYBOARD FLAG.
KRS          /BRING THE CHARACTER INTO THE AC.
.
.
```

A more convenient method of accomplishing this in one instruction as illustrated by the following:

```
KSF       /WAIT FOR THE FLAG TO SET.
JMP .-1
KRB       /CLEAR THE FLAG AND GET THE CHARACTER.
.
.
```

Data Transfers

As seen from the example above, data is moved from a peripheral into the AC under program control. However, the input transfer may be an OR with the previous contents of the AC, or it may be a jam transfer, depending on the design of the peripheral; therefore, the user should consult the detailed IOT listing for the specific peripheral (given in Chapter 7).

Output transfers work in a similar manner. Data is loaded into the AC using the TAD instruction. Then an output IOT is given to move the data to the peripheral and (usually) to initiate some sort of operation. The flag is used slightly differently, but it serves much the same purpose. Consider the following example involving the Teletype printer:

```
.
TAD DATA    /GET A CHARACTER TO BE PRINTED
TLS         /SEND IT TO THE PRINTER ELECTRONICS, CLEAR
            /THE FLAG, AND SAY "GO."
TSF         /HANG AROUND UNTIL THE FLAG SETS, SO YOU
JMP .-1     /DON'T ACCIDENTALLY TRY TO PRINT TWO
.           /DIFFERENT CHARACTERS AT THE SAME TIME.
.
```

If the printer flag is set ahead of time, the flag may be tested before doing the "TLS." This technique speeds up the program, since the computer can be doing instructions instead of just waiting for the flag.

```
TPF                    /SET PRINTER FLAG
.
.                      /GET CHARACTER
TSF
JMP .-1
TLS                    /PRINT IT
.
```

The user should not try to be exotic and microcode skip and clear IOTs into one instruction. It won't always work. Trouble could develop if the flag happened to set between the skip and clear functions, because the program might have cleared the flag before it found out the flag had been set.

PRINCIPLES OF PROGRAM INTERRUPTS

GENERAL

There may come a time when waiting for flags consumes too much valuable time. What is needed is some way of ignoring peripherals until they need attention.

For instance, the device flag sets when a new character is available if the device is an input device. If the device is an output device, the flag sets when the device is ready to receive a new character. Wouldn't it be desirable to service peripherals only if one or more flags happen to set?

To accomplish this, all peripherals OR their flags onto a special line called the Interrupt Request line. If this line becomes ground, it means that there's a flag (and hence a device) somewhere that needs attention. However, the flag must **demand** attention; not just pull a line which can be tested with an IOT. The "demand attention" circuitry is known as the interrupt system, and works as follows:

1. The interrupt system is automatically turned off when the computer is first turned on.
2. The interrupt system is turned on by an IOT (ION), but the actual enabling is delayed by one instruction. (The reason for this will be shown later.)
3. If the interrupt is enabled and some device's flag gets set, the processor **automatically** responds by executing a JMS to location zero and simultaneously turning off the interrupt system. Note that this JMS is hardware-generated.
4. In case it is needed, there's an IOT (IOF) which turns off the interrupt system.

The interrupt system is a simple piece of hardware, but has far-reaching program implications. Let's examine some of those implications.

Coding a Program Interrupt. Assume, for example, that the operation is reading and punching information using the PC8/E Reader/Punch combination. Between reading and punching, the processor is doing a simple, unrelated program. Once things get started, the procedure that follows is indicated in figure 5-1.

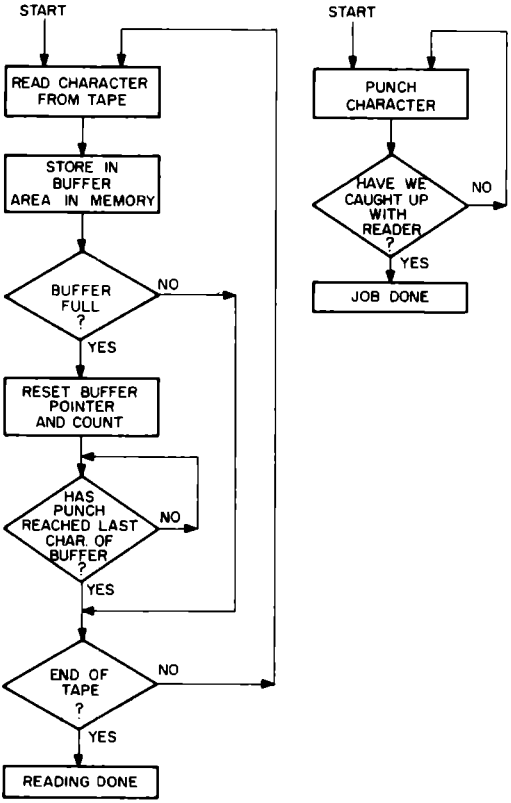


Figure 5-1 Program Interrupt Example

```
/The Interrupt Service Routine

0,      xxxx        /WHERE THE RETURN ADDRESS IS
                    /STORED WHEN THE INTERRUPT OC-
                    /CURS.
1,      JMP I.+1    /THIS INSTRUCTION GETS TO THE
2,      FLAGS       /FLAG TEST ROUTINE WHICH IS
                    /USUALLY ON OTHER THAN PAGE 0.

/The flag-testing and restoration routine

FLAGS,  DCA AC      /SAVE AC
        RAL
        DCA LNK     /AND LINK
        RSF         /CHECK READER FLAG
```

```
        SKP
        JMP RDR
        PSF             /AND PUNCH FLAG
        SKP
        JMP PUNCH
        KCC             /IF WE GET HERE, SOMEONE PROB-
                        /ABLY STRUCK A TELETYPE KEY
        TCF             /OR SOME OTHER STRANGE INTER-
                        /RUPT EXISTS.
DISMIS, CLA CLL         /HERE'S HOW WE EXIT
        TAD LNK         /RESTORE LINK
        RAR
        TAD AC          /AND AC,
        ION             /TURN ON THE INTERRUPT
        JMP I 0         /AND USE THE ONE-CYCLE DELAY
                        /TO GET THE PC LOADED.

/THE READER SERVICE ROUTINE. WE KNOW THE FLAG IS SET AL-
/READY, OR WE WOULDN'T BE HERE.

RDR,    RRB             /GET CHARACTER FROM READER,
                        /CLEAR FLAG
        DCA I 10        /SAVE IN BUFFER
        ISZ COUNT1
        JMP RDGO        /BUFFER NOT FULL
        TAD BUFF        /BUFFER FULL
        DCA 10
        TAD KCOUNT
        DCA COUNT1
        JMP DISMIS      /NOTICE-WE DID NOT MOVE
                        /READER.
RDGO,   RFC             /MOVE READER
        JMP DISMIS      /AND EXIT

/THE PUNCH SERVICE ROUTINE

PUNCH,  TAD I 11        /GET WORD FROM BUFFER.
        PLS             /PUNCH IT AND CLEAR FLAG
```

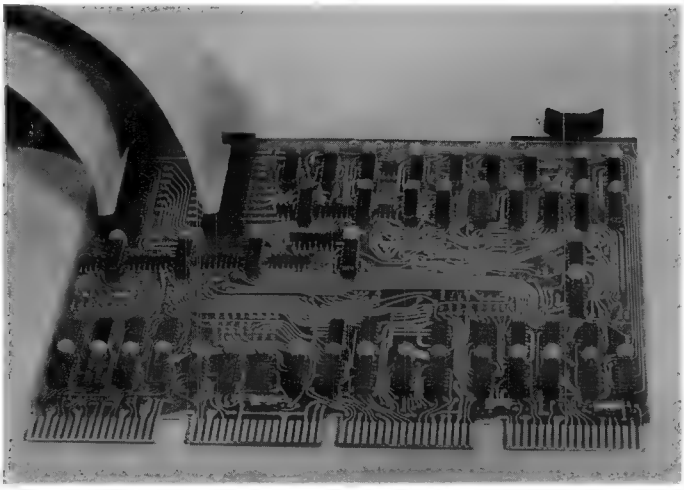
```
                CLA
                TAD 11          /COMPARE ADDRESS PUNCHED
                CIA
                TAD 10          /WITH READER ADDRESS
                SNA CLA
                JMP FINISH      /ADDRESSES EQUAL, SO JOB DONE.

                ISZ COUNT2
                JMP DISMIS
                TAD KCOUNT      /TOP OF BUFFER, SO RESET
                DCA COUNT2      /COUNTER
                TAD BUFF
                DCA 11          /AND POINTER.
                JMP RDGO        /AND RESTART THE READER.
/OK—THAT TAKES CARE OF THE INTERRUPT HANDLER. NOW COMES
/THE MAIN PROGRAM.
START,          RFC             /START READER
                PLS             /AND PUNCH
                TAD KCOUNT      /INITIALIZE ALL VARIABLES
                DCA COUNT1
                TAD KCOUNT
                DCA COUNT2
                TAD BUFF
                DCA 10
                TAD BUFF
                DCA 11
                ION             /AND TURN ON INTERRUPT
                ISZ PHONEY      /THIS IS A TRIVIAL NULL
                JMP .—1         /JOB WHICH SLOWLY INCREMENTS
                IAC             /THE AC.
                JMP .—3
FINISH,         HLT
/CONSTANTS
```

```
KCOUNT,     -100
BUFF,       377
/VARIABLES
COUNT1,     0
COUNT2,     0
AC,         0
LNK,        0
PHONEY,     0
```

The reader stops automatically when tape runs out of the reader. When the punch pointer becomes equal to the reader pointer, the job is finished. Notice how the one-instruction delay built into the ION instruction allows the computer to obtain the return address from location 0 before new interrupts can occur.

In the above example, the background null job (the job being done when interrupts were not being serviced) was unrelated to the interrupt routine. In the more general case, the background job is some sort of processing job operating on the buffer after reading and before punching. Clearly, the programming problem is more severe. The reader must be ahead of the processing, which must be ahead of the punching. Such problems are beyond the scope of this chapter, and the reader should, therefore, consult Chapter 5 of "Introduction To Programming."



External Data Break Interface
(Type KD8/E)

CHAPTER 6

DATA BREAK

GENERAL

Data break (sometimes called Direct Memory Access or DMA) is another form of data transfer used with high speed devices. Generally, a mass-storage device utilizes this form of data transfer. In the case of a high-speed device such as a disk, it becomes desirable to transfer a block of information at the fastest possible transfer rate. A block of data containing up to 4K words would require one data break for each word transferred.

Data break is the process of stealing memory cycles for the purpose of adding data to memory or removing data from memory without disturbing the major registers within the processor. Data breaks, therefore, stall the processor only during that period of time when some action is required by the processor, such as transferring data in or out of memory.

Data break is desirable when a large quantity of words is to be transferred. Transferring one 12-bit word is a relatively easy task to accomplish. If a word is to be transferred out to some peripheral, the only requirement would be to stop the program and jump to some subroutine that would address memory and transfer data out to that peripheral. Therefore, all that would be needed is a program subroutine, a memory address, and a 12-bit register to receive the 12-bit word at the peripheral end. It would obviously be more practical to use a programmed type of data transfer rather than a data break transfer.

To use data break transfers effectively, a series of words should be transferred and the peripheral should be considered a high-speed peripheral. To accomplish this, however, the process becomes more complicated. For example, if 50 words are to be transferred, 50 addresses are necessary, along with a means of telling the processor when the 50 words have been transferred. Thus, an address incrementing device is required to add a number to the current address and a word count mechanism is required to count the number of transfers.

THE BASIC DATA BREAK SYSTEM

The purchaser of a PDP-8/E and a data break peripheral receives, in addition to his basic programming package, subroutines corresponding to the data break device and a Programmer's Reference Manual to initially set up and program the device. Such subroutines might include the Interrupt Service Routine, the Search Subroutine, and the Read and Write Subroutines, depending upon the type of data break peripheral. These are first loaded into memory to generate the necessary IOTs required for addressing and initializing transfer control. The user then calls these subroutines.

The basic data break system is illustrated in Figure 6-1. The user calls for the data break, and the software routine initializes the data break transfer. The peripheral control contains a break priority circuit, a current address register, a word count register, and a data register.

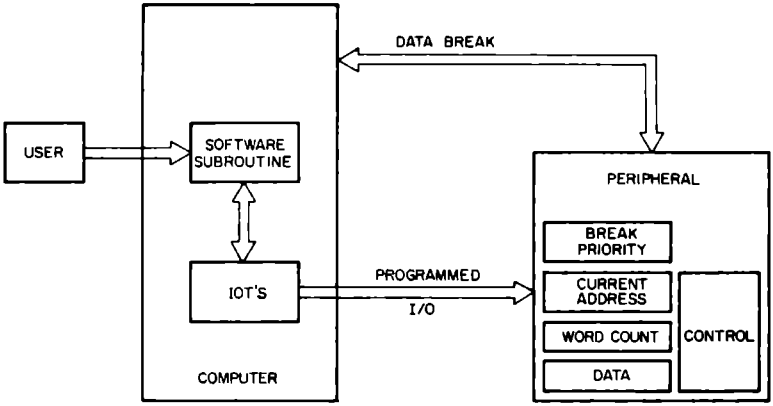


Figure 6-1 General Block Diagram of Basic Data Break Interfaces

Its function is to address memory, count the number of word transfers and receive or transmit data.

Current Address (CA) Register
The Current Address Register provides the ability to sequentially address a series of memory locations by incrementing before each transfer. Initially, the CA register is loaded with an address that is one location before the desired address. The register is then incremented (a one is added to the register) and the address is placed onto the memory data lines.

Word Count (WC) Register
The Word Count Register serves to count the number of data words in a block of data that is transferred from or to a peripheral. The two's complement of the number of words to be transferred is placed in the register initially. The register increments at each word transfer until the WC register contains zeros. When the WC register overflows, a signal is generated which clears the enable circuits.

Data Break Priority
Up to 12 data break devices can be simultaneously attached to the PDP-8/E. Each device is assigned a line on the OMNIBUS for use in determining priority.

Bit 0 represents the highest priority, and bit 11 represents the lowest. When a device makes a break request, the priority of the other devices also making a break request is tested. This guarantees that a faster and higher priority device will make the first data transfer.

Data Register
Data transfer occurs between the device Data Register and the Memory Buffer Register. The data register only serves either to receive or to transmit data.

Data Break Configurations

Two types of data break configurations (single-cycle and three-cycle) may be used on the PDP-8/E.

Single-cycle devices contain the WC and CA registers in the peripheral. Once data transfer begins, the processor stalls only once (one cycle is stolen) while the data transfer takes place.

Three-cycle devices use memory to contain the WC and CA registers. The processor then must stall one cycle for the word count, one cycle for the current address, and one cycle for the word transfer.

ONE-CYCLE DATA BREAK TRANSFERS

A simplified block diagram of the one-cycle data break operation is illustrated in Figure 6-2, and a flow chart showing the interaction between the processor and the device is illustrated in Figure 6-3. Because this is a one-cycle data break device, Figure 6-2 shows the word count and current address registers in the peripheral.

Figure 6-3 divides the chart in terms of actions required by the processor and actions required by the device. The "flow of events" time periods are divided to reflect the INITIAL SET-UP, DATA TRANSFER, and EXIT.

Initial Set-Up

The Program Subroutine initializes the word count (WC) and current address (CA) hardware registers located in the peripheral and generates IOTs to enable the device control logic and specify the direction of transfer.

Data Transfer

The 12-bit word count register becomes incremented just prior to a word transfer and provides an overflow signal to the control logic after the register is incremented to all zeros.

The 12-bit current address register is incremented at the same time as the WC register. If an overflow in the WC register occurs, the deivce clears the enable circuits after the current word has been transferred. Before the current address can be placed on the memory address lines, a priority test must be made. If the device is the highest priority, the contents of the CA register are placed on the memory address lines.

After the priority test, the break device generates CPU disabling signals which tell the processor to stall while a word is transferred into or out of memory. While the contents of the CA registers are placed on the MA lines, a memory direction signal line (transfer into or out of memory) is enabled. The transfer of data occurs between the memory buffer register and the device data register.

At the end of the memory cycle, the CPU disabling signals are removed and the processor resumes operation. If the device is still enabled (WC overflow has not occurred), it then repeats the data transfer cycle. If WC overflow has occurred, the device begins an EXIT.

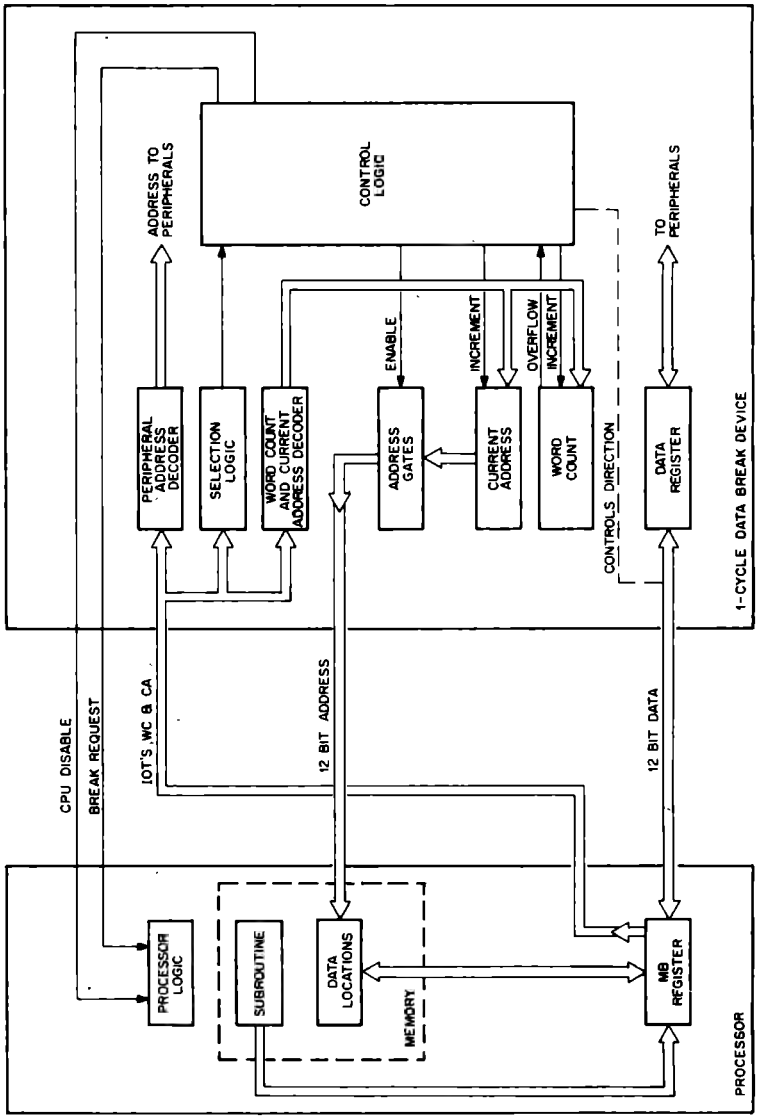


Figure 6-2 1-Cycle Data Break Simplified Block Diagram

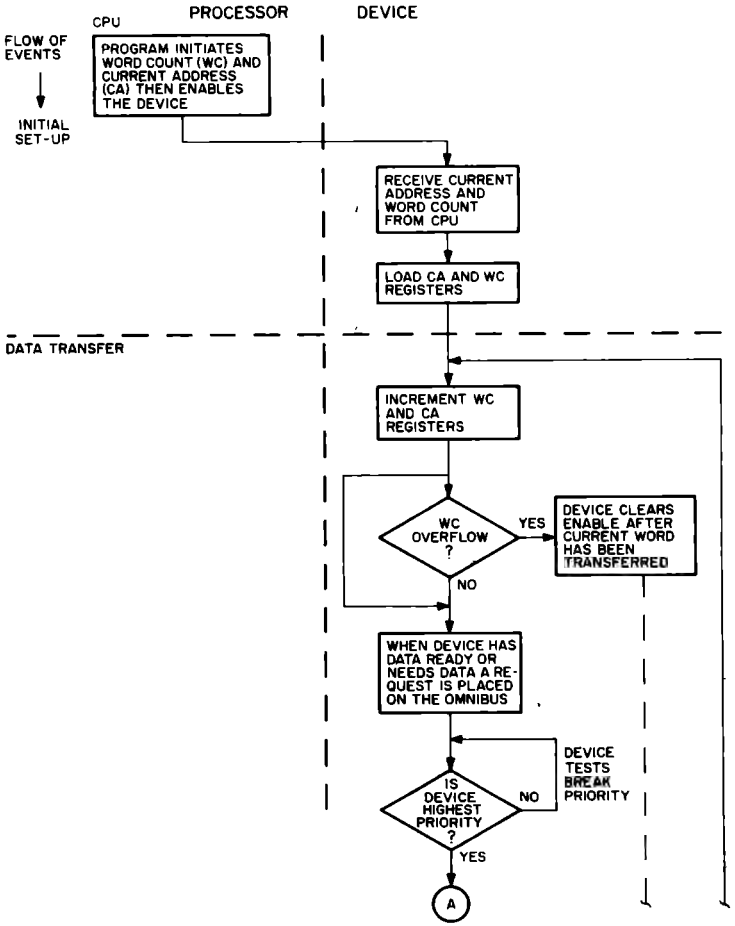

CPU
PROCESSOR
DEVICE
FLOW OF EVENTS
INITIAL SET-UP
PROGRAM INITIATES WORD COUNT (WC) AND CURRENT ADDRESS (CA) THEN ENABLES THE DEVICE
RECEIVE CURRENT ADDRESS AND WORD COUNT FROM CPU
LOAD CA AND WC REGISTERS
DATA TRANSFER
INCREMENT WC AND CA REGISTERS
WC OVERFLOW ?
YES
DEVICE CLEARS ENABLE AFTER CURRENT WORD HAS BEEN TRANSFERRED
NO
WHEN DEVICE HAS DATA READY OR NEEDS DATA A REQUEST IS PLACED ON THE OMNIBUS
DEVICE TESTS BREAK PRIORITY
IS DEVICE HIGHEST PRIORITY ?
NO
YES
A

Figure 6-3 1-Cycle Data Break Flow Chart

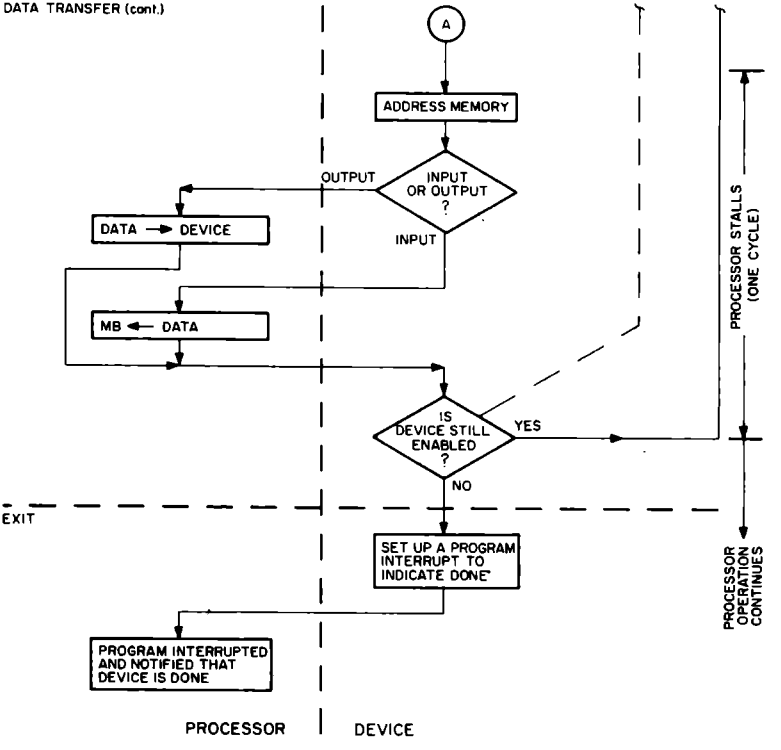


Figure 6-3 1-Cycle Data Break Flow Chart (Continued)

EXIT
To exit, the device sets a flag indicating that the block transfer has been completed. The processor then responds to this flag in the manner described in Chapter 5.

THREE-CYCLE DATA BREAK TRANSFERS
A simplified block diagram of the three-cycle data break operation is shown in Figure 6-4, and a flow chart illustrating the interaction between the processor and the device is shown in Figure 6-5. The WC and CA registers are located in memory. Therefore, the program loads the WC register with the required word count and the CA register with the address where the data will be either stored or retrieved.

The flow chart in Figure 6-4 is divided into time periods to reflect the INITIAL SET-UP, WORD COUNT, CURRENT ADDRESS, DATA TRANSFER, and EXIT. Because the word count and current address registers are located in memory, two additional data break cycles are required. Incrementing of both the CA and WC registers is accomplished in the processor.

Initial Set-Up
For a three-cycle operation, the data break subroutine must load the word count and current address memory locations. In addition, IOTs to enable the data break control logic and peripheral addresses are generated by the subroutine and transferred to the device.

Once the device decodes the IOTs, a break request is initiated and a priority test is performed. If the device has the highest priority, it generates a group of CPU disable signals and the system begins a word count cycle.

Word Count
The memory address of the word count register is hard-wired in the peripheral. This address is gated into the break address register and placed onto the memory address lines. An overriding line to the address register forces a zero into bit 11 (for an even address).

The processor then fetches the contents of the word count register to the memory buffer register. A one generated by the control logic is forced into bit 11 of the data bus, and transferred to the memory buffer register via the adder circuits. The resulting addition is tested for overflow. The contents of the memory buffer register are immediately placed into the word count register. If word count overflow occurred, the device clears its enable after the current word has been transferred. Just prior to entering the current address cycle, the priority is tested again.

Current Address
Updating the Current Address Register is accompilshed in the same manner as Word Count. However, instead of a zero being forced into bit 11 of the break address register, a one is used.

At the end of the current memory cycle, the contents of the MB register are written back into the CA register and also transferred to the device break address register. On the next memory cycle, the device break

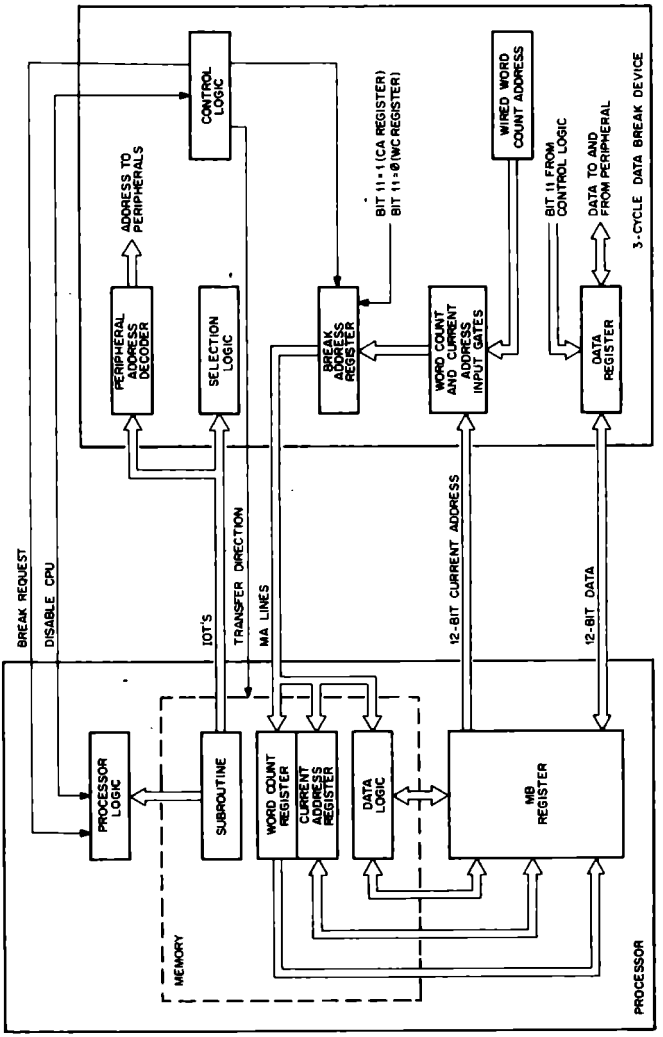


Figure 6-4 3-Cycle Data Break Simplified Block Diagram

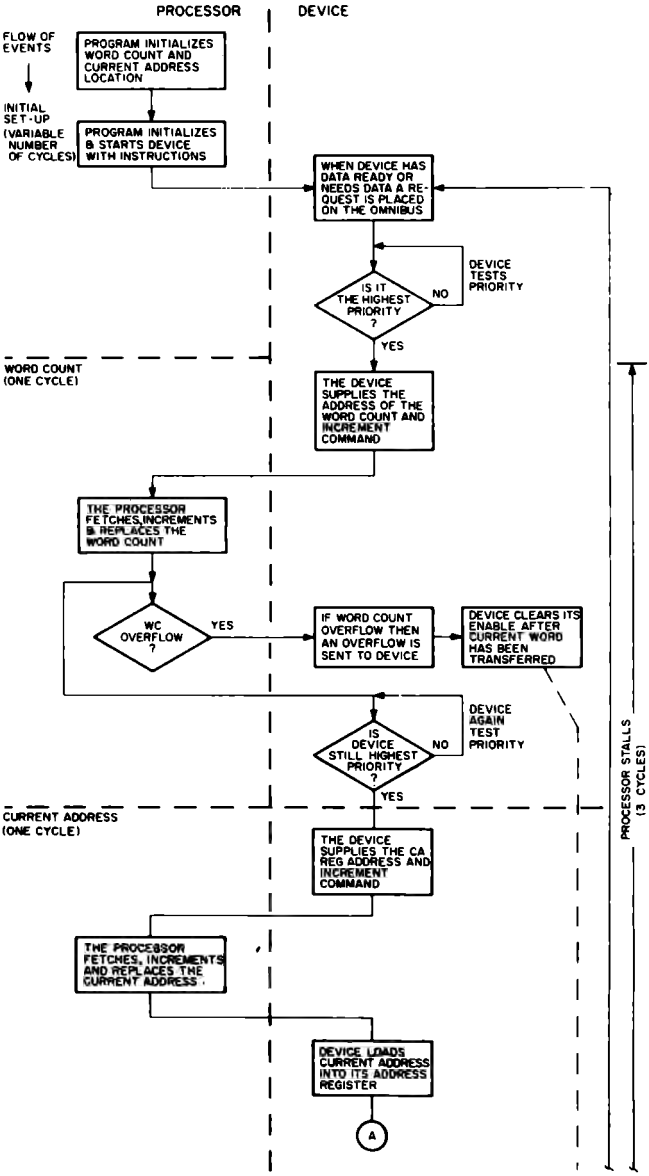

PROCESSOR
DEVICE
FLOW OF EVENTS
PROGRAM INITIALIZES WORD COUNT AND CURRENT ADDRESS LOCATION
INITIAL SET-UP (VARIABLE NUMBER OF CYCLES)
PROGRAM INITIALIZES & STARTS DEVICE WITH INSTRUCTIONS
WHEN DEVICE HAS DATA READY OR NEEDS DATA A REQUEST IS PLACED ON THE OMNIBUS
IS IT THE HIGHEST PRIORITY ?
NO
DEVICE TESTS PRIORITY
YES
WORD COUNT (ONE CYCLE)
THE DEVICE SUPPLIES THE ADDRESS OF THE WORD COUNT AND INCREMENT COMMAND
THE PROCESSOR FETCHES, INCREMENTS & REPLACES THE WORD COUNT
WC OVERFLOW ?
YES
IF WORD COUNT OVERFLOW THEN AN OVERFLOW IS SENT TO DEVICE
DEVICE CLEARS ITS ENABLE AFTER CURRENT WORD HAS BEEN TRANSFERRED
IS DEVICE STILL HIGHEST PRIORITY ?
NO
DEVICE AGAIN TEST PRIORITY
YES
CURRENT ADDRESS (ONE CYCLE)
THE DEVICE SUPPLIES THE CA REG ADDRESS AND INCREMENT COMMAND
THE PROCESSOR FETCHES, INCREMENTS AND REPLACES THE CURRENT ADDRESS.
DEVICE LOADS CURRENT ADDRESS INTO ITS ADDRESS REGISTER
A
PROCESSOR STALLS (3 CYCLES)

Figure 6-5 3-Cycle Data Break Flow Chart

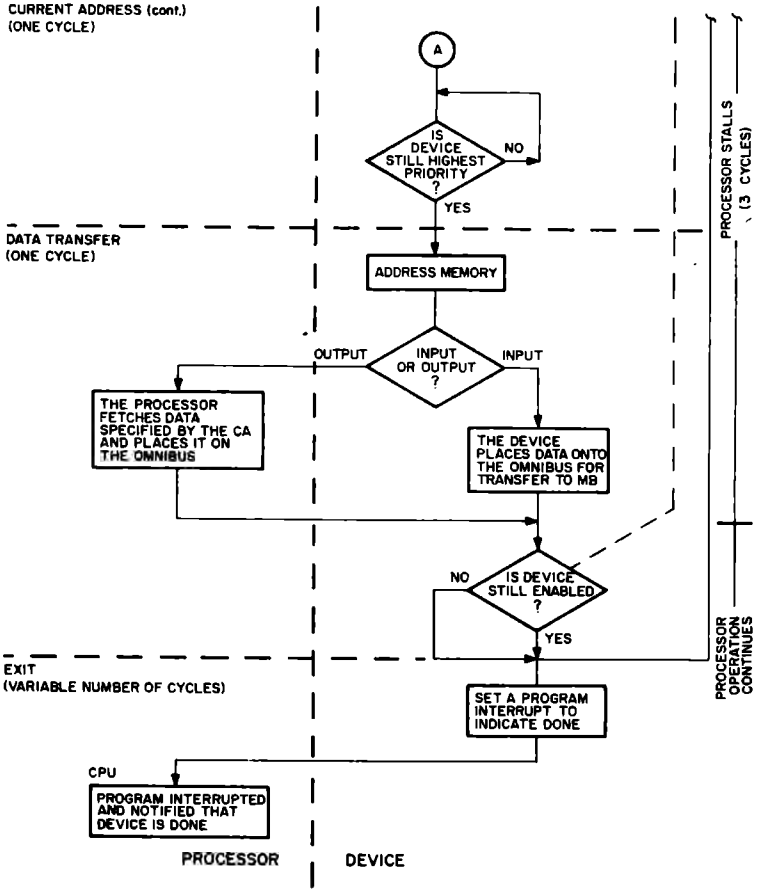


Figure 6-5 3-Cycle Data Break Flow Chart (Continued)

address register points to the data address and the system is now ready to transfer data. Just prior to addressing the data location, another priority test is made.

Data Transfer

When the current address is placed on the address lines, a transfer direction signal is generated, and data is placed onto the OMNIBUS data lines. The data transfer is between the device data register and the memory buffer register. The CPU disabling signals are removed at the end of the cycle. If the WC overflow signal has not been received, the device is still enabled. A break request is again initiated and the word count—current address—data transfer cycle is repeated. A word count overflow causes the device to set its device flag and to begin an exit.

Exit
A flag checking subroutine terminates the data break.

PROGRAMMING EXAMPLE
An example of programming for the data break operation is given below. This example deals with programming a DF32 Disk, and provides a subroutine only to transfer a block of words between Disk and Memory. It does not include the subroutines which are necessary to determine the word count, starting address in memory, and disk address. This example serves to illustrate the software required with data break.

BLOCK TRANSFER SUBROUTINE
Subroutine Call

```
•
•
•                       /PRECEDING INSTRUCTION OF THE MAIN
                        /PROGRAM
JMS I DRW               /DRW CONTAINS ADDRESS OF DISK ROUTINE
DEA & MEA               /CONTENTS TO BE PLACED IN THE DISK
                        /AND MEMORY EXTENDED ADDRESSES
CA                      /CONTENTS TO BE PLACED IN THE CURRENT
                        /ADDRESS REGISTER
WC                      /CONTENTS TO BE PLACED IN THE WORD
                        /COUNT REGISTER
MEMDIR                  /CONTENTS CONTAINS A 2 FOR READ AND 4
                        /FOR WRITE
DA                      /CONTENTS TO BE PLACED IN THE DISK
                        /ADDRESS
•                       /NEXT INSTRUCTION IN MAIN PROGRAM
•
•
•
DRW, DISK
DISK,   0               /RETURN POINTER
        TAD I DISK      /TRANSFER CONTENTS TO BE PLACED IN
                        /DEA and MEA TO AC REGISTER
        ISZ DISK        /MOVE POINTER TO NEXT LOCATION OF CALL
                        /(CA)
        DEAL            /LOAD DISK EXTENDED ADDRESS
        CLA             /CLEAR AC
        TAD I DISK      /TRANSFER CONTENTS TO BE PLACED IN CA
                        /REGISTER TO THE AC REGISTER
        ISZ DISK        /MOVE POINTER TO NEXT LOCATION (WC)
        DCA I DISKCA    /STORE UPDATED CURRENT ADDRESS
        TAD I DISK      /TRANSFER THE CONTENTS TO BE PLACED
                        /IN THE WC REGISTER TO AC
        ISZ DISK        /MOVE POINTER TO NEXT LOCATION (MEM
                        /DIR)
        DCA I DISKWC    /STORE UPDATED WORD COUNT
        TAD I DISK      /TRANSFER THE CONTENT OF MEM DIR (2
                        /OR 4) TO AC
        ISZ DISK        /MOVE POINTER TO NEXT LOCATION (DA)
```

```
        TAD IOT         /ADD 6601 TO THE AC TO BUILD READ OR
                        /WRITE IOT
        DCA GO          /DEPOSIT IOT TO BE EXECUTED
        TAD I DISK      /TRANSFER DISK ADDRESS TO AC
        ISZ DISK        /MOVE POINTER TO THE NEXT MEMORY
                        /LOCATION (MAIN PROGRAM)
GO,     0               /EXECUTE IOT TO LOAD DISK ADDRESS
                        /REGISTER AND BEGIN TRANSFER
        DFSC            /SKIP ON COMPLETION FLAG
        JMP .-1         /CHECK FLAG AGAIN
        DFSE            /SKIP ON NO ERROR FLAG
        JMP ERR         /JUMP ERROR
        JMP I DISK      /DISK CONTAINS ADDRESS OF NEXT
                        /INSTRUCTION OF MAIN PROGRAM
DISKCA, 7751
DISKWC, 7750
IOT,    6601
```

part 2

PDP-8/E OPTIONS

CHAPTER 7

PDP-8/E OPTIONS

SECTION 1 MECHANICAL EXPANSION OPTIONS
- SYSTEM EXPANDER BOXES
- PANEL OPTIONS

SECTION 2 COMPUTER INTERNAL OPTIONS
- EXTENDED ARITHMETIC ELEMENT
- MEMORY EQUIPMENT
- REAL TIME CLOCKS
- POWER FAIL DETECT

SECTION 3 OMNIBUS INPUT/OUTPUT EQUIPMENT OPTIONS
- CONSOLE TELETYPE
- PAPER TAPE READERS
- CRT DISPLAYS
- XY PLOTTERS
- LINE PRINTERS
- DATA COMMUNICATIONS EQUIPMENT
- CARD READERS
- LABORATORY PERIPHERALS
  - A-TO-D CONVERTERS
  - ANALOG MULTIPLEXERS
  - BUFFERED DIGITAL I/O

SECTION 4 EXTERNAL BUS INPUT/OUTPUT EQUIPMENT OPTIONS
- EXTERNAL BUS INTERFACE CONTROL
- RANDOM ACCESS DISK DEVICES
- MAGNETIC TAPE EQUIPMENT
- DATA ACQUISITION
  - A-TO-D CONVERTERS
  - ANALOG MULTIPLEXERS
  - INTEGRATING DIGITAL VOLTMETER
  - D-TO-A CONVERTERS
  - UNIVERSAL DIGITAL CONTROLLER

CHAPTER 7

PDP-8/E OPTIONS

GENERAL

Chapter 7 contains descriptions of all the standard peripheral devices that are optionally available for the PDP-8/E computer. Section 1 deals with the mechanical expansion options. Section 2 covers the computer internal options. Section 3 describes the OMNIBUS input/output equipment options. Section 4 is concerned with the external bus input/output equipment options.

SECTION 1 MECHANICAL EXPANSION OPTIONS

Included in this section are those options which affect the external physical properties of the PDP-8/E computer, such as cabinets and panels. Further details regarding installation of these options appear in Chapter 11 of this handbook.

SYSTEM EXPANDER BOXES

Type BA8-AA System Expander Box

The BA8-AA includes a power chassis assembly and OMNIBUS assembly, capable of accommodating up to 20 PDP-8/E modules, and a BC08H-3F Cable Set (three and a half feet in length) with rack-mountable slides included, as well as a Type KC8-EB blank front panel.

Type BA8-AB System Expander Box

The BA8-AB includes a power chassis assembly and OMNIBUS assembly, capable of accommodating up to 20 PDP-8/E modules, and a BC08H-3F Cable set (three and a half feet in length) with table-top cover included, as well as a Type KC8-EB blank front panel.

Type BE8-A OMNIBUS Expander

The BE8-A includes an additional OMNIBUS assembly, capable of accommodating up to 20 PDP-8/E modules, together with M935 Bus Connectors for expanding either the PDP-8/E, the BA8-AA, or the BA8-AB to 38 slots.

PANEL OPTIONS

Type KC8-EC Turn-Key Front Panel

The KC8-EC panel contains a key-lock ON/OFF switch for controlling the application of power to the PDP-8/E system. The KP8-E Power Fail option is a prerequisite for using this panel. It is used as an alternate panel to the PDP-8/E.

Type KC8-EB Blank Front Panel

This panel can be used in place of the KC8-EA panel or the KC8-EC panel on the PDP-8/E, and is supplied with the BA8-AA or BA8-BA Expander Box option. It is used on the PDP-8/E when the system is equipped with a KP8-E, and the user wishes to switch power externally.

SECTION 2 COMPUTER INTERNAL OPTIONS

This section deals with internal options for the PDP-8/E computer, including those concerned with the extended arithmetic facility, memory equipment, real-time measurement, and power fail detection and restart.

Many of these are covered in greater detail in Chapters 3 and 4 of this handbook.

The execution time for IOT instructions in this section is 1.2 μs, except where otherwise specified in the instructions for extended arithmetic.

Type KE8-E Extended Arithmetic Element

The KE8-E option for the PDP-8/E enables the central processor to perform arithmetic operations at high speeds by incorporating the EAE components with the existing central processor logic so that they operate asynchronously. The option consists of circuits that perform parallel arithmetic operations on positive binary numbers, and includes:

a. A 5-bit step counter (SC) register—This register is loaded with the complement of the contents of MB(7-11) for the ASR, LSR, SCL, and SHL instructions, is loaded from AC 7-11 for the ACS instruction, and is set to contain all zeros for the DVI, MUY, and NMI instructions. It is used to record the number of shifts performed, and stops the shifting process after the correct number of shifts.

b. A 4-bit instruction register (EAE IR)—This register consists of flip-flops set to MB(6, 8-10) during the Fetch cycle of an EAE instruction.

c. The EAE timing and control logic—EAE logic is contained in modules which plug into the OMNIBUS. The KE8-E EAE logic circuits are used in conjunction with the accumulator (AC), link (L), multiplier quotient (MQ), and memory buffer (MB), though asynchronously with them, to perform arithmetic operations. When this option is added to a PDP-8/E system, a class of instructions is added to the Group 3 Operate instruction list.

Programming

The Extended Arithmetic Element (EAE) microinstructions are specified by an operate instruction (operation code 7) in which MB(3) and MB(11) always contain binary 1s.

At printing time, the EAE option for the PDP-8/E was still in the design stage, so that the following instructions are subject to change. The first group includes those instructions available in the basic PDP-8/E system:

No Operation (NOP)

Octal Code: 7401
Execution Time: 1.2 μs
Operation: No operation is performed.

Load Multiplier Quotient (MQL)

Octal Code: 7421
Execution Time: 1.2 μs
Operation: Transfers the content of the AC to the MQ, and clears the AC.

Multiplier Quotient Load into Accumulator (MQA)

Octal Code: 7501
Execution Time: 1.2 μs
Operation: Inclusive ORs the content of the MQ with the AC and transfers the result to the AC.

Clear Accumulator (CLA)

Octal Code: 7601
Execution Time: 1.2 μs
Operation: Clears the AC.

Clear Accumulator and Multiplier Quotient (CLA, MQL)

Octal Code: 7621
Execution Time: 1.2 μs
Operation: Clears the AC and MQ.

Clear Accumulator, Load Multiplier Quotient into Accumulator (CLA, MQA)

Octal Code: 7701
Execution Time: 1.2 μs
Operation: Transfers the content of the MQ to the AC.

Load Multiplier Quotient into Accumulator, Clear Multiplier Quotient (CLA, MQL, MQA)

Octal Code: 7721
Execution Time: 1.2 μs
Operation: Clears AC, loads content of MQ into AC, and clears MQ.

Swap Accumulator and Multiplier Quotient (SWP)

Octal Code: 7521
Execution Time: 1.2 μs
Operation: Interchanges the contents of the AC and MQ.

The following instructions are available only with the KE8-E Extended Arithmetic Element option.

Store Accumulator in Step Counter (ACS)

Octal Code: 7403
Execution Time: 1.2 μs
Operation: Transfers the contents of AC(7-11) to the SC, and clears the AC.

OR Step Counter with Accumulator (SCA)

Octal Code: 7441
Execution Time: 1.2 μs
Operation: Inclusive ORs the content of SC with AC, and transfers the result to the AC.

Load Accumulator with Step Counter (CLA, SCA)

Octal Code: 7641
Execution Time: 1.2 μs
Operation: Transfers content of SC to AC(7-11).

Subtract Accumulator from Multiplier Quotient (SAM)

Octal Code: 7453
Execution Time: 1.2 μs
Operation: Subtracts content of AC from MQ and places result in AC. Sets GT flag if MQ is equal to or greater than original AC; clears GT flag if MQ is less than original AC. Sets L if there is a carry or borrow; clears L if there is neither.

Skip if Greater Than (SGT)

Octal Code: 6006
Execution Time: 1.2 μs
Operation: Skips the next instruction if the GT flag is set.

Clear Step Counter and Accumulator (CLA, ACS)

Octal Code: 7603
Execution Time: 1.2 μs
Operation: Clears SC and AC.

Shift Left (SHL)

Octal Code: 7413
Execution Time: 2.6 μs + 0.3N μs (where N = number of shifts)
Operation: Shifts the combined contents of AC and MQ to the left the number of positions specified in the next core location following this instruction. Treats L, AC, and MQ as one register; bits shifted from AC go to L, bits shifted from L are lost, and zeros are shifted into MQ(11).

Arithmetic Shift Right (ASR)

Octal Code: 7415
Execution Time: 2.6 μs + 0.3N μs (where N = number of shifts)
Operation: Shifts the combined contents of AC and MQ to the right the number of positions specified in the next core location following this instruction. Treats the L, AC, and MQ as one long register; MQ(11) is shifted into L, and bits shifted from L are lost. The sign bit, contained in AC(0) is saved in AC(0) and also shifted into the vacated AC position with each shift.

Logical Shift Right (LSR)

Octal Code: 7417
Execution Time: 2.6 μs + 0.3N μs (where N = number of shifts)
Operation: Shifts the combined contents of AC and MQ to the right the number of positions specified in the next core location following this instruction. Zeros enter the vacated AC(0) position, MQ(11) is shifted into L, and bits shifted from L are lost.

Multiply (MUY)

Octal Code: 7405
Execution Time: 7.4 μs
Operation: Multiplies MQ by the number whose address is in the next core location following this instruction. Sets L to zero. The most significant 12 bits of product are contained in AC; the least significant 12 bits are contained in MQ.

Divide (DIV)

Octal Code: 7407
Execution Time: 7.7 μs
Operation: Divides the 24-bit dividend (12 most significant bits in AC, 12 least significant bits in MQ) by the number whose address is in the next core location following this instruction. Places the quotient in the MQ and the remainder in the AC. The link is zero unless divide overflow occurred, in which case the operation would be canceled after the first cycle of the division.

Normalize (NMI)

Octal Code: 7411
Execution Time: 1.2 μs + 0.3N μs (where N = number of shifts)
Operation: Converts a binary number to a fraction and exponent for use in floating-point arithmetic. Shifts the combined contents of the AC and the MQ left until AC(0) does not equal AC(1), or until 6000 0000 is contained in the combined AC and MQ. Zeros are shifted into vacated MQ(11) positions for each shift, with AC(0) going into the link. Data shifted from the link is lost. If 4000 0000 (negative zero) is contained in the combined AC and MQ initially, the AC is cleared. At the conclusion of this instruction, the step counter contains a number equal to the number of shifts performed.

Absolute Normalize (NMU)

Octal Code: 7451
Execution Time: 1.2 μs + 0.3N (where N = number of shifts)
Operation: Shifts the combined contents of the AC and the MQ left until AC(0) = 1. Zeros are shifted into vacated MQ(11) positions for each shift, AC(0) is shifted into the link, and data shifted out of the link is lost. At the conclusion of this instruction, the step counter contains a number equal to the number of shifts performed.

Double Precision Two's Complement (DCM)

Octal Code: 7567
Execution Time: 1.8 μs
Operation: Computes the two's complement of the combined contents of the AC and the MQ, and places it in the AC and MQ.

Double Precision Add (DAD)

Octal Code: 7563
Execution Time: 5.2 μs
Operation: Adds a 24-bit number (least significant 12 bits from an address in the next core location following this instruction, most significant 12 bits from that address +1) to the 24-bit number held in AC and MQ (least significant 12 bits from the MQ, most significant 12 bits from the AC). If there is a carry out, the link is set; if not, the link is cleared.

Double Precision Store (DST)

Octal Code: 7565
Execution Time: 5.2 μs
Operation: Stores the MQ in the address contained in the core location following this instruction, stores the AC in that address + 1, and clears the AC and MQ.

Double Precision Increment (DPIC)

Octal Code: 7575
Execution Time: 2.0 μs
Operation: Increments the combined contents of AC and MQ (with least significant bits in MQ). If there is a carry, link = 1; if not, link = 0. Previous contents of link are lost.

Double Precision Skip if Zero (DPSZ)

Octal Code: 7457
Execution Time: 1.2 μs
Operation: Skips the next instruction if the contents of the AC and the MQ are both zero. Those contents are not affected by this instruction.

Being augmented instructions, the EAE instructions are microprogrammable, and can be combined with one another to perform nonconflicting logical operations, as indicated in Figure 7-1.

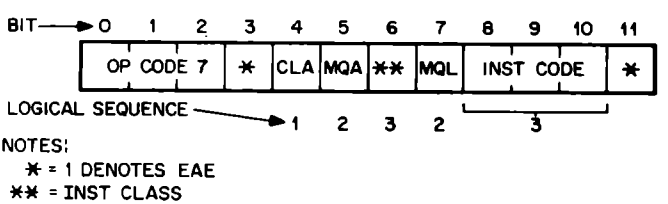


Figure 7-1 EAE Microinstruction Bit Assignments

MEMORY EQUIPMENT OPTIONS

The basic memory (MM8-E) is a 4096-word, random-access core memory that performs all the functions of data storage and retrieval. The MM8-E is packaged on three PDP-8/E modules that plug into the OMNIBUS. These modules, when used with the KM8-E Memory Extension and Time Share option, can extend memory capacity up to 32,768 words in increments of 4096 words. In addition, this option enables the PDP-8/E to operate in a time-sharing environment. With the addition of the MP8-E Memory Parity option, all transfers to and from memory can be checked for parity. Other memory options include the MR8-EA 256-Word Read-Only Memory, the MR8-EB 1024-word Read-Only Memory, the MW8-E 256-word Read/Write Memory, and the MI8-E Bootstrap Loader.

KM8-E Memory Extension and Time-Share Option

This option provides the user with two primary capabilities. The memory extension portion extends the addressing capabilities of the machine from 4069 words up to 32,768 words. The time-share portion enables the computer to operate in either the normal manner (Executive Mode) or the User Mode. User Mode enables the machine to function in a time-sharing environment in which a user program is prevented from disturbing or interfering with another user program. The KM8-E option is packaged on one PDP-8/E module that plugs into the OMNIBUS. This option is required whenever memory capacity is extended beyond 4096 words.

Memory Extension Description

The functional circuit elements which make up the memory extension control perform as follows:

Instruction Field Register (IF)—The IF is a three-bit register that serves as an extension of the PC. The contents of the IF determine the field from which all instructions are taken and the field from which operands are taken in directly-addressed AND, TAD, ISZ, or DCA instructions. Depressing the console EXTD ADDR LOAD switch transfers the instruction field in SWITCH REGISTER bits 6 through 8 into the IF register. During a JMP or JMS instruction, the IF is set by transfer of information from the instruction buffer register. When a program interrupt occurs, the contents of the IF are automatically stored in bits 0 through 2 of the save field register for restoration to the IF from the instruction buffer register at the conclusion of the program interrupt subroutine.

Data Field Register (DF)—This three-bit register determines the memory field from which operands are taken in indirectly-addressed AND, TAD, ISZ, or DCA instructions. Depressing the console EXTD ADDR LOAD switch transfers the SWITCH REGISTER bits 9 through 11 into the DF register. During a CDF instruction, the DF register is loaded from MD6-8 to establish a new data field. When a program interrupt occurs, the contents of the DF are automatically stored in bits 3-5 of the save field register. The DF is set by a transfer of information from save field register bits 3 through 5 by the RMF instruction. This action is required to restore the data field at the conclusion of the program interrupt subroutine.

Instruction Buffer Register (IB)—The IB serves as a three-bit input buffer for the instruction field register. All field number transfers into the instruction field register are made through the instruction buffer, except transfers from the operator's console switches. The IB is set by depressing of the console EXTD ADDR LOAD switch in the same manner as the instruction field register. A CIF microinstruction loads the IB with the programmed field on MD6-8. An RMF microinstruction transfers save field register bits 0 through 2 into the IB to restore the instruction field that existed before a program interrupt.

Save Field Register (SF)—When a program interrupt occurs, this seven-bit register is loaded from the user build flip-flop, and the IR and UF registers. The SF register is loaded during the cycle in which the program count is stored at address 0000 of the JMS instruction forced by a program interrupt request, then the instruction field and data field registers are cleared. An RMF instruction can be given immediately before exit from the program interrupt subroutine to restore the instruction field and data field by transferring the SF into the IB and the DF registers. (Also, see GTF and RTF instructions.)

Extended Address Gating—This logic consists of an output gating structure and control logic for gating the extended memory field address to core memory. The contents of the IF register are placed on the EMA0-2 lines unless an AND I, TAD I, ISZ I, or DCA I instruction is encountered. If such an instruction is encountered, the contents of the IF are placed on EMA 0-2 for the Fetch and Defer cycles, and the contents of the DF are placed on EMA0-2 for the Execute cycle. The extended memory field address is changed only at TP4 and remains available for the entire memory cycle.

Data Transfer Gating—This gating allows the contents of the save field register, instruction field register, or the data field register to be strobed into the accumulator via DATA lines 6-11. During an RIB or GTF instruction, bits 6 through 11 of the AC receive contents of the save field register. During an RIF instruction, bits 6 through 8 of the AC receive the contents of the instruction field register. During an RDF instruction, bits 6 through 8 of the AC receive the contents of the data field register.

Device Selector and Instruction Decoding—Bits 3 through 5 of the IOT instruction are decoded to produce the IOT command pulses for the memory extension control. Bits 6 through 8 of the instruction are not used for device selection since they specify a field number in some commands. Therefore, the select code for this device selector is designated as 2N. Bits 9 through 11 are also decoded to implement specific commands. The instruction decoding logic is common to the time-share portion of the KM8-E option.

Programming

Instructions associated with the extended memory portion KM8-E option are defined below:

Get Flags (GTF)

Octal Code: 6004
Operation: Reads the contents of the interrupt inhibit flip-flop, and the SF register to AC3, AC5-11 respectively. The other AC bits are loaded with information from the CPU and the EAE; i.e., link, greater-than-flag, interrupt bus, interrupt on.

Restore Flags (RTF)

Octal Code: 6005
Operation: Loads the user buffer flip-flop, the instruction buffer register, and the data field register with the contents of AC bits 5, 6-8, and 9-11 and inhibits processor interrupts until the next JMP or JMS instruction. At the conclusion of the JMP or JMS instruction, the contents of the user buffer flip-flop and the instruction buffer register are transferred into the user field flip-flop and the instruction field register, respectively. The contents of the other AC bits are loaded into the CPU and EAE to cause the converse of the GTF instruction. The Interrupt On flip-flop in the CPU is unconditionally set by this instruction.

Change to Data Field N (CDF)

Octal Code: 62N1
Operation: Loads the data field register with the program-selected field number (N = 0 to 7). All subsequent memory requests for operands are automatically switched to that data field, except for directly-addressed AND, TAD, ISZ, or DCA instructions.

Change to Instruction Field N (CIF)

Octal Code: 62N2
Operation: Loads the instruction buffer register with the program-selected field number (N = 0 to 7) and inhibits processor interrupts until the next JMP or JMS instruction. At the conclusion of either of these instructions, the contents of the instruction buffer register are transferred into the instruction field register.

Change Data Field, Change Instruction Field (CDF, CIF)

Octal Code: 62N3
Operation: Performs the combination of CDF and CIF operations.

Read Data Field (RDF)

Octal Code: 6214
Operation: ORs the contents of the data field register into bits 6-8 of the AC. All other bits of the AC are unaffected.

Read Instruction Field (RIF)

Octal Code: 6224
Operation: ORs the contents of the instruction field register into bits 6-8 of the AC. All other bits of the AC are unaffected.

Read Interrupt Buffer (RIB)

Octal Code: 6234
Operation: ORs the contents of the save field register (which is loaded from the instruction and data field during a program interrupt) into bits 6-8 and 9-11 of the AC, respectively. Thus, AC 6-11 contains the instruction and data fields that were in use before the last program interrupt. AC 5 is loaded by the time-share bit of the save field register. All other bits of the AC are unaffected.

Restore Memory Field (RMF)

Octal Code: 6244
Operation: Restores the contents of the save field register (which is loaded from the instruction and data field during a program interrupt) into the instruction buffer, the data field register, and the user buffer (if time share option is enabled). This command is used upon exit from the program interrupt subroutine in another field.

Instructions and data are accessed from the currently assigned instruction and data fields, where instructions and data may be stored in the same or different memory fields. When indirect memory references are executed, the operand address refers first to the instruction field to obtain an effective address, which, in turn, refers to a location in the currently assigned data field. All instructions and operands are obtained from the field designated by the contents of the instruction field register, except for indirectly addressed operands, which are specified by the contents of the data field register. In other words, the DF is effective only in the execute cycle that is directly preceded by the defer cycle of a memory reference instruction, as follows:

Indirect (Bit 3)	Page or Z Bit (Bit 4)	Field In IF	Field In DF	Effective Address
0	0	m	n	The operand is in page 0 of field m at the page address specified by bits 5 through 11.
0	1	m	n	The operand is in the current page of field m at the page address specified by bits 5 through 11.
1	0	m	n	The absolute address of the operand in field n is taken from the contents of field m located in page 0 designated by bits 5 through 11.

1	1	m	n	The absolute address of the operand in field n is taken from the contents of field m located in the current page, designated by bits 5 through 11.

Each field of extended memory contains eight auto-index registers in addresses 10 through 17. For example, assume that a program in field 2 is running (IF = 2) and using operands in field 1 (DF = 1) when the instruction TAD I 10 is fetched. The defer cycle is entered (bit 3 = 1), and the contents of location 10 in field 2 are read, incremented, and rewritten. If address 10 in field 2 originally contained 4321, it now contains 4322. In the execute cycle, the operand is fetched from location 4322 of field 1. Program control is transferred between memory fields by the CIF instruction. The instruction does not change the instruction field directly, as this would make it impossible to execute the next sequential instruction; instead, it loads the new instruction field in the IB for automatic transfer into the IF when either a JMP or JMS instruction is executed. The DF is unaffected by the JMP and JMS instructions.

The 12-bit program counter is set in the normal manner and, because the IF is an extension on the most significant end of the PC, the program sequence resumes in the new memory field following a JMP or JMS. Entry into a program interrupt is inhibited after the CIF instruction until a JMP or JMS is executed.

NOTE

The IF is not incremented if the PC goes from 7777 to 0000. This feature protects the user from accidentally entering a nonexistent field.

To call a subroutine that is out of the current field, the data field register is set to indicate the field of the calling JMS, which establishes the location of the operands as well as the identity of the return field. The instruction field is set to the field of the starting address of the subroutine. The following sequence returns program control to the main program from a subroutine that is out of the current field.

```
/PROGRAM OPERATIONS IN MEMORY FIELD 2
/INSTRUCTION FIELD = 2; DATA FIELD = 2
/CALL A SUBROUTINE IN MEMORY FIELD 1
/INDICATE CALLING FIELD LOCATION BY THE CONTENTS OF THE DATA
 FIELD

        CIF 10          /CHANGE TO INSTRUCTION
                        /FIELD 1 = 6212
        JMS I SUBRP     /SUBRP = ENTRY ADDRESS
        CDF 20          /RESTORE DATA FIELD
          .
          .
          .
SUBRP,  SUBR            /POINTER
                        /CALLED SUBROUTINE, LOCATED IN
                        /FIELD 1
```

```
SUBR,   0               /RETURN ADDRESS STORED HERE
        CLA
        RDF             /READ DATA FIELD INTO AC
        TAD RETURN      /CONTENTS OF THE AC = 6202 +
                        /DATA FIELD BITS
        DCA EXIT        /STORE INSTRUCTION SUBROUTINE
          .             /NOW CHANGE DATA FIELD IF DESIRED
          .
          .
          .
EXIT,                   /A CIF INSTRUCTION
        JMP I SUBR      /RETURN TO CALLING PROGRAM
RETURN, CIF             /USED TO CALCULATE EXIT
                        /INSTRUCTION
```

When a program interrupt occurs, the current instruction and data field numbers are automatically stored in the 6-bit save field register, then the IF and DF are cleared. The 12-bit program count is stored in location 0000 of field 0 and program control advances to location 0001 of field 0. At the end of the program interrupt subroutine, the RMF instruction restores the IF and DF from the contents of the SF. Alternatively, the GTF and RTF instructions may be used to handle the Save Field and Link information. The following instruction sequence at the end of the program interrupt subroutine continues the interrupted program after the interrupt has been processed:

```
          .             /RESTORE MQ IF REQUIRED
          .
          .
          .             /RESTORE L IF REQUIRED
          .
          .
          .
          .
        CLA
        TAD AC          /RESTORE AC
        RMF             /LOAD IB FROM SF
        ION             /TURN ON INTERRUPT SYSTEM
        JMP I 0         /RESTORE PC WITH CONTENTS OF
                        /LOCATION 0 AND LOAD IF FROM IB
OR
0,      0               /PC STORAGE
        DCA ACSV        /SAVE AC,
        MQA CLA
        DCA MQSV        /MQ,
        GTF
        DCA FLAGS
          .
          .
          .
        CLA
        TAD MQSV
```

```
MQL             /RESTORE MQ
TAD FLAGS
RTF             /REPLACE FLAGS, ION
CLA
TAD ACSV        /AC
JMP I 0         /AND EXIT
```

Time-Share Description

The Time-Share portion of the KM8-E operates in two modes as denoted by the user flag (UF) flip-flop. When the UF flip-flop is in the logic 1 state, operation is in the user mode and a user program is running in the central processor. When the UF flip-flop is in the logic 0 state, operation is in the executive mode and the time-sharing system's monitor is in control of the central processor. The four instructions (CINT, SINT, CUF, and SUF) are used by the time-sharing system's monitor in the executive mode and are never used by a user program. If a user program attempted to use one of these instructions, execution of the instruction would be blocked (see next paragraph). The KM8-E option adds the necessary hardware to the PDP-8/E to implement these instructions.

In executive mode, the computer operates normally. When the computer is operated in user mode, operation is normal except for IOT, HLT, LAS, and OSR instructions. When one of these instructions is encountered, the hardware inhibits the normal instruction sequence (other than rewriting the instruction in memory), and generates an interrupt at the end of the current memory cycle by setting the UINT flip-flop. The time-sharing system's monitor program then analyzes the source of interrupt, and takes appropriate action.

The time-share option requires at least 8K of core memory; thus, it is packaged with the memory extension option. A jumper on the KM8-E module is used to select the time-share function. The module is shipped with this jumper in place (time-share function disabled).

Programming

Instructions associated with the time-share portion of the KM8-E are defined as follows:

Clear User Interrupt (CINT)

Octal Code: 6204
Operation: Clears the user interrupt flip-flop.

Skip on User Interrupt (SINT)

Octal Code: 6254
Operation: Increments the PC when the user interrupt flip-flop is set so the next sequential instruction is skipped.

Clear User Flag (CUF)

Octal Code: 6264
Operation: Clears the user buffer flip-flop.

Set User Flag (SUF)

Octal Code: 6274
Operation: Sets user buffer flip-flop and inhibits processor interrupts until the next JMP or JMS instruction. At the conclusion of either of these instructions, the content of the user buffer flip-flop is transferred into the user field flip-flop.

MP8-E Memory Parity

The memory parity option adds the circuits required to generate, store, and check the parity of memory words. This option replaces the 12-bit memory system with, effectively, a 13-bit system by adding the generating and storage capabilities for the parity bit. Odd parity (odd number of binary ones in the 13-bit word) is generated and stored for each word entered into memory. Parity is formed for each word retrieved from memory and this result is checked against its stored parity bit. If the two differ, a parity error flag is set to indicate that an error occurred. This flag is normally connected to the program interrupt system to cause the computer to enter a program interrupt subroutine for locating the interrupt source. Once the interrupting source is located, the computer enters an appropriate service routine to service the error condition. This routine can repeat the program step in which the error occurred to verify the error condition, can perform a simple read/write check for the error's address, or can determine machine status for the error detected and re-establish or print out these conditions, and then halt. The routine can also return the machine to the main program.

The MP8-E option consists of three PDP-8/E modules that plug into the OMNIBUS. Two of these modules (X-Y Driver and Current Source, and Core Stack) are identical to those of the MM8-E basic core memory and use the same addressing methods. However, only eight bits of the possible 12 bits are used. These eight-bit locations correspond to the eight possible memory fields and store up to 32,768 (8×4096) parity bits. The third module (Sense-Inhibit) contains device and operation decoding circuits, field decoding circuits, eight sense amplifiers, an eight-bit register, eight inhibit drivers and circuits for controlling the operations. This module also contains three control and status flip-flops that are controlled by IOT instructions. These flip-flops select odd or even parity generation and checking, enable or disable interrupts for parity errors, and store a parity error condition.

The following routine initializes the parity bits for a read-only or write-protected memory:

```
                        /INITIALIZE LOC 10 WITH STARTING ADD.
                        /TURN OFF PARITY INTERRUPT
                        /SET COUNTER
LOOP,   TAD I 10        /READ DATA, REWRITE PARITY
        ISZ COUNT
        JMP LOOP        /CONTINUE UNTIL DONE
                        /CLEAR PARITY ERROR FLAG
                        /TURN ON PARITY INTERRUPT
```

Programming
Instructions associated with the MP8-E option are:

Disable Memory Parity Error Interrupt (DPI)

Octal Code: 6100
Operation: Disables the generation of interrupts for parity errors by clearing the interrupt enable flip-flop of the memory parity option.

Skip On No Memory Parity Error (SMP)

Octal Code: 6101
Operation: Senses the memory parity error flag; if it contains a 0 (signifying no error has been detected), the PC is incremented so that the next instruction is skipped.

Enable Memory Parity Error Interrupt (EPI)

Octal Code: 6103
Operation: Enables interrupts from the memory parity option. The memory parity interrupt is automatically enabled when power is turned on, by the CLEAR key on the front panel and by the CAF IOT instruction.

Clear Memory Parity Error Flag (CMP)

Octal Code: 6104
Operation: Clears the memory parity error flag. The parity error flag is also cleared when power is turned on, by the CLEAR key on the front panel, and by the CAF IOT instruction.

Skip on No Memory Parity Error and Clear Memory Parity Error Flag (SMP, CMP)

Octal Code: 6105
Operation: Senses the memory parity error flag; if it contains a 0, the next instruction is skipped. The memory parity error flag is then cleared.

Check For Even Parity (CEP)

Octal Code: 6106
Operation: Causes parity to be checked for an even number of binary 1's in the entire word. This operation is effective only during the execute cycle immediately following this instructon.

Skip on Memory Parity Option (SPO)

Octal Code: 6107
Operation: Increments the PC when the system includes a memory parity option so that the next sequential instruction is skipped.

Use of these instructions is discussed below:

a. The DPI instruction is useful in certain diagnostic maintenance programs where it is desired to disable interrupts resulting from parity errors. This instruction also gives the user more flexibility for multiple program interrupt usage.
b. The SMP instruction is used as a programmed check for memory parity errors. When used in a program interrupt subroutine, this instruction can be followed by a jump to a portion of the routine that services the memory parity option.
c. The EPI instruction is used to return the memory parity option to normal operation after a DPI command.
d. The CMP instruction initializes the memory parity option in preparation for normal programmed operation of the computer.
e. The CEP instruction is useful in diagnostic maintenance programs. By altering the parity check from odd to even, parity errors can be forced, to permit checking for proper functions of the parity option.
f. The SPO instruction permits the user to automatically check whether or not the system is equipped with a memory parity option.
g. The SMP, CMP instruction is a combination of SMP and CMP instructions, and permits the operations performed by these instructions to be implemented by one instruction.

MW8-E 256-Word Read/Write Memory

This option is a 256-word read/write memory with a write-protect feature. The MW8-E option can be configured with the basic core and is required whenever a MR8-E ROM is used. When used with the basic 4096 core memory, a KM8-E Memory Extension and Time-Share option is also required.

The MW8-E option has four primary uses:

a. It is used with ROM for variable storage and interrupt handling.
b. It is used to simulate a ROM (through write-protect feature) for program debugging and short-term storage.
c. It is used as a general-purpose ROM such as might be needed for program constants.
d. It can be used with the time-sharing feature of KM8-E to protect a user monitor program.

The MW8-E option is packaged on two PDP-8/E modules that plug into the OMNIBUS. One module contains X-Y drive circuits and the core memory. The second module contains the sense-inhibit and input/output gating circuits.

The write-protect feature is selected using a switch mounted on the sense-inhibit module. When placed to ROM, this switch prevents writing of input data into the storage. When placed to NORM, both read and write operations can be performed.

An MW8-E can be assigned any memory field address; however, it must be assigned a block of 256 addresses beginning with an even-number

page. Field and page addresses are selected using jumpers in its address decoding circuits. When used in other than field 0, the KM8-E option is also required.

MR8-EA 256-Word Read-Only-Memory
The MR8-EA option provides the user with read-only-memory (ROM) capabilities such as might be used for hardwired controller, communications or process-control functions. This option is provided in 256-word increments package on one module. However, the module, because of its thickness, requires two module slots.

Information stored by the ROM is established by wiring of the unit at the factory. Therefore, the information content must be specified by the user at the time of purchase. A recommended approach to defining the ROM content is to use the MW8-E 256-Word Read/Write option to simulate the ROM (through use of the MW8-E write-protect feature) for program definition and debugging. A copy of the resulting program can then be supplied to define the content of the ROM.

The number of ROM modules used is limited only by the amount of basic core or read/write capability required and the maximum address capabilities of the machine. A ROM can be assigned any memory field address, however, it must be assigned a block of 256 addresses beginning with an even-number memory page. Field and page addresses are selected by jumpers on its address decoding circuits. When used in other than field 0, the KM8-E option is required.

In situations where a small amount of ROM is desired, an MR8-E can be installed which uses locations already allotted to the 4K memory. The MR8-E automatically disables core memory using the same address. The core addresses can be re-enabled by removal of the MR8-E.

MR8-EB 1024-word Read-Only Memory
(same as MR8-EA, except larger)

MI8-E Bootstrap Loader
This option uses a 32-word read-only-memory (ROM) with diodes that can be arranged to accommodate any program up to 32 words in length. This option is normally used as a hardware Read-In-Mode (RIM) paper tape loader for loading of programs from the PDP-8/E paper tape reader of the console teleprinter. However, it can be used for any user-designated programs of 32 words or less. The MI8-E option is contained on one PDP-8/E module that plugs into the OMNIBUS.

The MI8-E operates in a shadow address mode with core memory. That is, the addresses used for this device can overlap core memory addresses and can be used by core memory whenever the MI8-E option is not operating. The MI8-E can be used in any memory field; the field is selected by jumpers on the module. For a 32-word program, the MI8-E occupies the last 32 locations in the field (7740 (octal) through 7777 (octal)). The starting and ending addresses within this 32-address group are selected by jumpers on the module. Thus, programs requiring less than 32 locations can also be readily implemented.

The MI8-E option is selected, using the console SW control. However, this control has no effect unless the machine is stopped (RUN flop is reset). When this control is depressed, addresses 7740 (octal) through 7777 (octal) access the MI8-E hardware only. Core memory is prevented from responding to these addresses by outputs of the MI8-E control logic.

To operate the MI8-E option, the SW key is depressed, loading the starting address and starting the computer. The MI8-E then assumes control and provides instructions from its ROM to the MD lines during each FETCH major state. These instructions can load paper tape programs from the PDP-8/E paper tape reader or the console teleprinter, or perform user-designated functions. When the ending address is reached (as determined by module jumpers and MA inputs), the last instruction is executed and the Bootstrap Loader resets itself.

Real Time Clock Options

Type DK8-EA Real Time Clock (Line Frequency)

The DK8-EA is a fixed-interval line frequency clock option to the PDP-8/E that causes an interrupt 50 or 60 times per second, depending on line frequency. The clock and control are contained on one PDP-8/E module, which plugs into the OMNIBUS.

Programming

The following instructions control the DK8-EA line frequency clock:

Enable Interrupt (CLEI)

Octal Code: 6131
Operation: Enables the clock interrupt so that each clock pulse will cause a program interrupt request.

Disable Clock Interrupt (CLDI)

Octal Code: 6131
Operation: Disables the clock interrupt so that the clock cannot cause program interrupts.

Skip on Clock Flag and Clear Flag (CLSK)

Octal Code: 6133
Operation: Senses the clock flag, which is set with each clock pulse; if it is set, the next sequential instruction is skipped, and the clock flag is cleared.

Type DK8-EC Real Time Clock (Crystal)

The DK8-EC is a fixed-interval crystal-controlled clock option to the PDP-8/E that is used to cause an interrupt every 50, 500, or 5,000 times per second (jumper selectable). The clock frequency is derived from a 20-MHz crystal. The clock and control are contained on one PDP-8/E module, which plugs into the Omnibus.

Programming

The instructions which control the DK8-EC crystal clock are the same as those shown above for the DK-EC line frequency clock.

Type DK8-EP Programmable Real Time Clock

The DK8-EP real time clock option offers the PDP-8/E user a method for accurately measuring and counting intervals or events in a number of ways. It can be used to synchronize the central processor to external events, count external events, measure intervals of time between events, or provide program interrupts at programmable intervals. Some of these operations can be performed concurrently.

The DK8-EP system consists of:

a. A 12-bit binary counter using MSI integrated circuits with an overflow bit.
b. A 12-bit buffer register.
c. A 20-MHz crystal clock with frequency dividers.
d. A PDP-8/E module (M860) containing all control functions, IOT decoding, and registers.
e. A PDP-8/E module (M518) containing the input logic and Schmitt Triggers to make up three external input channels. This module and module M860 are joined together by an edge connector, type H851, which connects the contacts on the top edges of the boards. Option jumpers on the M518 connect the Schmitt Triggers to the input logic, permitting it to accept a wide variety of input signals from external sources.

Logically, the DK8-EP contains the following features:

a. Clock Enable Register
   This register controls the rate of the time base and the mode of counting, and selectively enables each of the three input channels and the interrupt line.
b. Clock Buffer
   The Clock Buffer stores data being transferred from the AC to the clock counter, or from the clock counter to the AC. It also permits presetting of the clock counter.
c. Clock Counter
   This register is a 12-bit binary counter that may load the clock buffer or be loaded from it. When an overflow occurs and the clock enable mode is 01, the clock buffer is automatically loaded into the clock counter. The overflow is set by the most significant bit of the clock enable register going from 1 to 0.
d. Programmable Time Base
   The Programmable Time Base provides count pulses to the clock counter according to the rate set by the clock enable register.
e. External Input Channels
   These three channels are used to actuate the clock, cause an interrupt, or transfer the contents of the clock counter to the clock buffer.
f. Crystal Clock
   The clock is a simple crystal-controlled clock, which operates at 20 MHz + or − 0.1%. MSI integrated circuit decade counters divide the base clock frequency down to any of the following rates: 1 MHz, 100 kHz, 10 kHz, 1 kHz, or 100 Hz.

Input Synchronizers

The three external input channels are used to convert external events into a synchronized control and status signal for the clock. Each input channel consists of an input Schmitt trigger with pulse generator, four flip-flops, and associated control gating. The Schmitt Trigger/pulse generator converts the preselected voltage threshold crossing by an external signal into a signal pulse. The control gating is set by the clock enable register. The three input channels provide for three separate input events to actuate the clock, interrupt, or transfer the content of the clock counter to the clock buffer, as well as the following capabilities:

a. An event at Channel 1 may be used as an input to the clock counter.
b. An event at Channel 2 may be used to cause an A/D start pulse for the Analog-to-Digital Converter, AD8-EA.
c. An event at Channel 3 may be used to reset the clock counter.

The Schmitt Triggers can accept pulse or continuously varying analog inputs. These inputs may be cabled directly to the M518 module, or the Schmitt Triggers may be connected to potentiometers and three conductor phone jack inputs on the DK8-EF Clock Control Panel.

There are four flip-flops associated with each input circuit:

Input Enable Flip-Flop—This flip-flop gates on and off input signals to the clock. It is set and cleared under program control.

Input Flip-Flop—This flip-flop is set by the external signal from the Schmitt Trigger input. It provides synchronization between external timing and internal clock timing.

Sync Flip-Flop—This flip-flop is loaded from the input flip-flop on the next Status to AC pulse, and remains set during succeeding Status to AC pulses. When this is loaded, the input flip-flop is cleared.

Enable Event Interrupt Flip-Flop—This flip-flop connects the input flip-flop the Interrupt line of the OMNIBUS and permits external events to cause program interrupts. It is set and cleared under program control via bit 8 in the clock enable register.

There are no operator controls on the DK8-EP, and no special maintenance instructions are necessary.

Schmitt Triggers Specifications

Nominal Input Voltage Range	+ or −5V
Input Type	Differential
Input Resistance	50 K
Minimum Duration Input Pulse	2μs
Maximum Permissible Input Voltage	+ or −50V
Hysteresis	0.3 V
Common Mode Rejection	35 db
Propagation	600 ns

Schmitt Triggers, when connected to Clock Control Panel DK8/EF

Input Threshold	Variable between + and −2v
Source	+, −, or line frequency zero crossing.

Programming

The following IOT instructions control the DK8-EP real time clock:

Skip on Clock Interrupt (CLSK)

Octal Code: 6131

Operation: Causes the content of the PC to be incremented by one if an interrupt condition exists, so that the next instruction is skipped. The interrupt conditions are as follows:

a. Enable Event Interrupt 1 and Input 1
b. Enable Event Interrupt 2 and Input 2
c. Enable Event Interrupt 3 and Input 3
d. Enable Overflow Interrupt and Overflow

AC to Clock Buffer (CLAB)

Octal Code: 6133

Operation: Causes the content of the AC to be transferred into the Clock Buffer; then causes the content of the Clock Buffer to be transferred into the Clock Counter. The AC is not changed.

Clear Clock Enable Register per AC (CLZE)

Octal Code: 6130

Operation: Clears the bits in the Clock Enable Register corresponding to those bits set in the AC. The AC is not changed.

Set Clock Enable Register per AC (CLDE)

Octal Code: 6132

Operation: Sets the bits in the Clock Enable Register corresponding to those bits set in the AC. The AC is not changed.

Load Clock Enable Register (CLEN)

Octal Code: 6134

Operation: Causes the content of the Clock Enable Register to be transferred into the AC.

Clock Enable Registers Functions

AC BIT	FUNCTION
0	Enables clock overflow to cause an interrupt.
1 & 2	Mode
	00 Counter runs at selected rate. Overflow occurs every 4096 counts. Flag remains set.

	01	Counter runs at selected rate. Overflow causes Clock Buffer to be transferred to the Clock Counter, which continues to run. Overflow remains set until cleared with IOT 6135.
	10	Counter runs at selected rate. When an enabled event occurs, the Clock Counter is transferred to the Clock Buffer, and the Counter continues.
	11	Counter runs at selected rate. When an enabled input occurs on channel 3, the Clock Counter is transferred to the Clock Buffer, and the Clock Counter continues to run from zero.
3, 4, & 5	Count Rate	
	111	Stop
	110	1 MHz
	101	100 kHz
	100	10 kHz
	011	1 kHz
	010	100 Hz
	001	External input from Channel 1 (Bit 11, which is the enable bit for Channel 1, does not have to be set.)
	000	Stop
6	Overflow starts ADC. (When the Clock Counter overflows, the analog-to-digital converter, type AD8-EA, is started.)	
7	Input starts ADC (An event occurring in Channel 2 causes the ADC, type AD8/EA, to be started if the channel is enabled.)	
8	Events in Channels 1, 2, or 3 cause an interrupt request.	
9, 10, & 11	Enable Events 1, 2, and 3	
	9 —Event 3	
	10—Event 2	
	11—Event 1	

Clock Status to AC (CLSA)

Octal Code: 6135

Operation: Interrogates the Clock Input and Overflow Status flip-flops. The clock status information is inclusively ORed into the AC, then the status bits corresponding to set AC bits are cleared. This ensures that only one occurrence of an Event will be transferred to the program. The status condition is established as follows:

AC Bit	Status Condition
0	Overflow
9	Event 3
10	Event 2
11	Event 1

Clock Buffer to AC (CLBA)

Octal Code: 6136
Operation: Clears the AC, then transfers the content of the Clock Buffer into the AC.

Clock Counter to AC (CLCA)

Octal Code: 6137
Operation: Clears the AC, transfers the content of the Clock Counter to the Clock Buffer, then transfers the content of the Clock Buffer into the AC.

NOTE

The clock counter may be read while it is counting. Gating in the clock control section prevents data from being strobed out of the counter before a specified time following a clock pulse. This time, approximately 300 ns, allows the data to settle in the counter.
This feature allows the counter to be read any number of times without introducing timing errors in counting the amount of time between intervals, and also eliminates false counts that are the result of reading the counter as one or more bits are in transition from one state to another.

Example Subroutine #1
This example illustrates how the DK8-EP can be used as a double-precision (24-bit) free-running clock, using the clock counter as the low order 12 bits and a memory location as the high order 12 bits. Because all of the clock's registers have been set to zero initially by the clear key, the program needs only to zero the high order words, set the enable register, and turn on the interrupt. After 4096 counts, the clock counter overflows, signalling an interrupt. The service routine simply increments the high order word, then returns to the main program.

```
        CLA
        DCA HIGH        /ZERO HIGH ORDER WORD
        TAD ENABLE      /OVERFL + MODE 00 + RATE
        CLOE            /SET ENABLE REGISTER
        ION             /INTERRUPT ON

ENABLE = OVERFL + MODE 00 + RATE

                        /SERVICE ROUTINE
        CLSK            /CLOCK SKIP?
        JMP OTHERS      /NOT A CLOCK FLAG
        CLSA            /READ STATUS, CLEAR FLAGS
        SPA CLA         /IGNORE OTHER CLOCK INTERRUPTS
        ISZ HIGH        /INCREMENT HIGH
        JMP RETURN      /RETURN TO MAIN PROGRAM
```

With this simple program, time can be kept during program execution. With the clock set to its fastest rate (1 μs per tick), this double-precision counter could mark time for only just over 16 seconds; with the clock set to its slowest rate, it could mark time for over 100 days.

A simple routine could be written to interrogate elapsed time by using the CLCA (clock counter to AC) command.

Example Subroutine #2
The DK8-EP can also easily be programmed to function as an alarm clock, counting off a period of time, giving an alarm, automatically re-setting itself, and continuing. The alarm could be used to ring a bell, as indicated in the example; however, a more practical use would be to start an analog-to-digital conversion to take a number of samples from the outside world.

This example will ring the bell every second:

```
START,    CLA
          TAD COUNTER        /SET COUNTER TO -1000
          CLAB
          CLA
          TAD ENABLE         /SET ENABLE REGISTER
          CLOE
AGAIN,    CLSK               /CLOCK SKIP?
          JMP .-1
          CLSA               /YES, READ STATUS
          CLA
          TAD BELL           /RING BELL
          TLS
          TSF
          JMP .-1
          JMP AGAIN
COUNTER,  -1750
ENABLE,   MODE 01 + 1 MS
BELL,     207
```

This program could easily be modified to work in the interrupt mode by setting bit 0 of the enable register to a 1. An interrupt would then occur every second, and this could be used to ring the bell.

Type KP8-E Power Fail Detect
The KP8-E and its related shut-down and restart subroutines are designed to restore computer operation automatically following a failure of the computer's primary power source. This OMNIBUS option protects an operating program in the event of such a failure by causing a program interrupt, enabling continued operation for 1 ms; this allows the interrupt routine to detect the low power as initiator of the interrupt and to store both the contents of active registers (AC, L, MQ, etc.) and the program count in known core memory locations.

Variations of the AC line below the predetermined threshold level at a rate of one per second or less will also cause the shut-down circuits to be activated. When power is restored the power low flag clears, and a routine beginning in address 0000 starts automatically. This routine

restores the contents of the active registers and program counter to the conditions that existed when the interrupt occurred, then continues the interrupted program.

The power failure option consists of three circuits, contained on a single PDP-8/E module.

a. A power interrupt circuit, which monitors the status of the computer power supply and sets a power low flag when power is interrupted (due to a power failure or to the operation of the POWER switch on the operator's console). This flag causes a program interrupt when an interruption in computer power is detected.

b. A shutdown sequence circuit, which ensures that, when a power interrupt occurs, the computer logic circuits will continue operation for 1 ms to allow a program subroutine to store the contents of the active registers. If, at the end of the 1 ms interval, computer operation still continues, it is halted. When power conditions are suitable for computer operation, a restart circuit clears the power low flag and restarts the program. A manual RESTART switch located on the right side of the power fall module enables or disables the automatic restart operation. With this switch in the ON (up) position, the option clears the MA and produces a MEMORY START pulse 500 ms after power conditions are satisfactory. The MA is cleared so that operation restarts by executing the instruction in address 0000. That instruction must be a JMP to the starting address of the subroutine that restores the contents of the active registers and the program counter to the conditions existing prior to the power low interrupt. The 500-ms delay ensures that slow mechanical devices, such as Teletype equipment, have completed any previous operation before the program is resumed. Simulation of the manual START function causes the processor to generate a power clear pulse to clear internal controls and I/O device registers. With the RESTART switch in the OFF (down) position, the power low flag is cleared, but the program must be started manually, possibly after resetting peripheral equipment or by starting the interrupted program from the beginning. The shut-down circuitry is unaffected by the switch.

c. A skip circuit provides programmed sensing of the condition of the power low flag by adding the IOT SPL (6102) instruction to the computer repertoire.

Programming
Skip On Power Low (SPL)

Octal Code: 6102
Operation: Senses the content of the power low flag. If the power low flag contains a 1 (indicating that a power failure has been detected), the contents of the PC are incremented by one, so that the next sequential instruction is skipped.

Because the time that computer operation can be extended after a power failure is limited to 1 ms, the condition of the power low flag should be the first status check made by the program interrupt subroutine. The interrupt subroutine, starting with the SPL microinstruction (and including the power fail program sequence), can be executed in less than 30 μs. The power fail program sequence stores the contents of the active registers and program counter in designated core memory locations, then relocates the calling instruction of the power restore subroutine to address 0000, as follows:

```
Address   Instruction     Remarks

0000      —               /STORAGE FOR PC AFTER PROGRAM IN-
                           TERRUPT
0001      JMP FLAGS       /INSTRUCTION EXECUTED AFTER PRO-
                           GRAM INTERRUPT

FLAGS,    SPL             /SKIP IF POWER LOW FLAG = 1
          JMP OTHER       /INTERRUPT NOT CAUSED BY POWER
                          /LOW, CHECK OTHER FLAGS
          DCA AC          /INTERRUPT WAS CAUSED BY POWER
                          /LOW, SAVE AC
          RAR             /GET LINK
          DCA LINK        /SAVE LINK

          MQA             /GET MQ
          DCA MQ          /SAVE MQ
          TAD 0000        /GET PC
          DCA PC          /SAVE PC
          TAD RESTRT      /GET RESTART INSTRUCTION
          DCA 0000        /DEPOSIT RESTART INSTRUCTION IN 0000
          HLT
RESTRT,   JMP ABCD        /ABCD IS LOCAION OF RESTART ROUTINE
```

Automatic program restart begins by executing the instruction stored in address 0000 by the power fail routine. The power restore subroutine restores the contents of the active registers, enables the program interrupt facility, and continues the interrupted program from the point at which it was interrupted, as follows:

```
Address   Instruction     Remarks

0000      JMP ABCD
ABCD,     TAD MQ          /GET MQ
          MQL             /RESTORE MQ
          TAD LINK        /GET LINK
          CLL RAL         /RESTORE LINK
          TAD AC          /RESTORE AC
          ION             /TURN ON INTERRUPT
          JMP I PC        /RETURN TO INTERRUPTED PROGRAM
```

SECTION 3 OMNIBUS INPUT/OUTPUT EQUIPMENT OPTIONS

This section describes those options to the PDP-8/E computer that perform transfers of information to and from the computer by means of the OMNIBUS. In all cases, execution time for IOT instructions in this section is 1.2 μs.

Many of these options and their manner of transfer are discussed in greater detail in Chapters 5 and 9 of this handbook.

CONSOLE TELETYPE

Model ASR 33 Teletype

The standard Teletype Model ASR 33 (automatic send-receive) is used to type in or print out information at a rate of up to ten characters per second, or to read in or punch out perforated tape at ten characters per second. Signals transferred between the Model ASR 33 and the control logic are standard, serial, 11-unit code, Teletype signals. The signals consist of spaces and marks which correspond to open circuit and bias current in the Teletype, and to Os and 1s in the Teletype control and computer. The start bit (space, 0, open circuit) and subsequent eight-character bits are one-unit-of-time duration and are followed by the stop bit, which occupies two units.

The eight-bit code used by the Model ASR 33 Teletype unit is the American Standard code for information interchange (ASCII) modified. To convert the ASCII code to Teletype code, add 200 octal (ASCII + 200 (octal) = Teletype). This code is read in the normal octal form used in the computer. Bits are numbered from right to left, from 1 through 8, with bits 1 through 3 containing the least significant octal number. The first information bit transmitted is bit 1, which is read into AC11. Figure 7-2 illustrates the context and description of the ASCII Teletype code and its associated bit content in the AC.

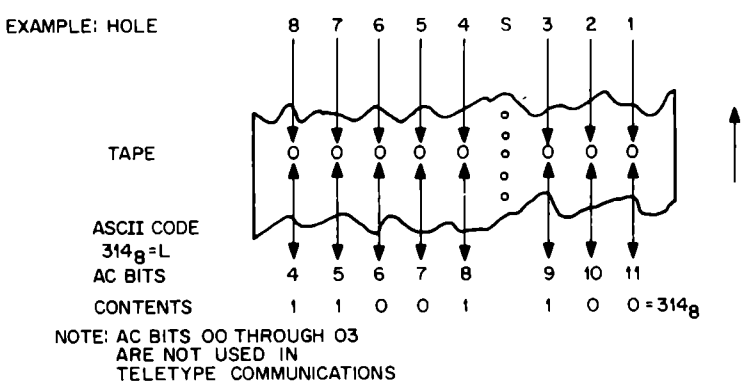

EXAMPLE: HOLE
8 7 6 5 4 S 3 2 1
TAPE
O O O O O O O O
ASCII CODE
314_8=L
AC BITS
4 5 6 7 8 9 10 11
CONTENTS
1 1 0 0 1 1 0 0 = 314_8
NOTE: AC BITS 00 THROUGH 03
ARE NOT USED IN
TELETYPE COMMUNICATIONS

Figure 7-2 Relationship Between Teletype Tape AC Contents and Binary and Octal Number Being Transferred

The ASR 33 generates all assigned codes except 340 through 374 and 376, which are not assigned. Generally, codes 207, 212, 240 through 337, and 377 are sufficient for Teletype operation. The ASR 33 detects all characters, but does not interpret all of the codes that it generates as commands.

The standard number of characters printed per line by the ASR 33 is 72. The sequence for proceeding to the next line is a carriage return followed by a line feed (as opposed to a line feed followed by a carriage return). Appendix C lists the Teletype character code. Punched tape format (for 264 (octal)) is as follows:

Binary Code			Tape Channel	
(Punch = 1)	8 7	6 5 4	S	3 2 1
Octal Code	1 0	1 1 0	(Sprocket)	1 0 0

KL8-E Console Teleprinter Control

The KL8-E control unit is a PDP-8/E module which plugs into the OMNIBUS and controls the operation of the Teletype from programmed instructions. This module contains the shift clock, the control logic for IOT decoding, parallel-to-serial and serial-to-parallel converters, and program control of interrupt and flag facilities. Serial information read or written by the Teletype unit is assembled or disassembled by the KL8-E control for transfer between the Teletype and the AC.

For program operation, the Teletype unit and control are considered as a Teletype In (TTI) for input intelligence from the keyboard or the perforated-tape reader, and as a Teletype Out (TTO) for computer output information to be printed and/or punched on tape. Therefore, two device select codes are used; select code 03 initiates operations associated with the keyboard/reader (TTI), and select code 04 performs operations associated with the teleprinter/punch (TTO). The control unit contains a programmable interrupt enable flip-flop that is common to both the keyboard/reader and teleprinter/punch. This flip-flop is set when power is turned on or INITIALIZE is generated, and can be set or cleared (as specified by AC11) using the KIE instruction. If AC11 is a 1 when the KIE instruction is issued, the interrupt enable flip-flop is set to permit the generation of interrupt requests whenever the keyboard/reader flag or teleprinter/punch flag is set. In contrast, if AC11 is a 0 when KIE is issued, no interrupt can be generated by this control unit. Functions performed by the keyboard/reader and teleprinter/punch are described in subsequent paragraphs.

Specifications

Interface	20 ma current loop operation. Reader controlled by reader enable leads
Character Parameters	8 data bits and 1 or 2 (standard) stop units
Baud Rate (Standard)	110 (Other rates available upon special order)
Binary Input/Output	8-bit parallel

Other features

1) Programmable interrupt enable
2) Program control for clearing keyboard flag without setting reader run flip-flop.
3) Program control for setting teleprinter flag.

Keyboard/Reader

The keyboard and tape reader control contains an eight-bit shift register (TTI) which assembles and holds the code for the last character struck on the keyboard or read from the tape. Teletype characters from the keyboard/reader are received serially by the TTI register. The teletype character code is loaded into the TTI so that spaces (the absence of holes) correspond with binary 0s and holes (marks) correspond to binary 1s. Upon program command, the contents of the TTI are transferred in parallel to the AC.

When a Teletype character is to be read from the paper tape reader, the control de-energizes a relay in the Teletype unit to release the tape feed latch. When released, the latch mechanism stops tape motion only when a complete character has been sensed, and before sensing of the next character is started.

When an eight-bit character has been assembled in the TTI, the keyboard flag is set to cause a program interrupt if the interrupt enable flip-flop has been set. When the program services the interrupt, it senses the flag with a KSF instruction and, with the flag set, issues a KRB instruction which clears the AC, clears the keyboard flag, transfers the contents of the TTI into the AC, and enables advance of the tape feed mechanism.

Programming

The following instructions are used for supplying data to the computer from the keyboard/reader:

Clear Keyboard Flag (KCF)

Octal Code: 6030
Operation: Clears the keyboard flag without setting the reader run flip-flop. The AC is not cleared by this instruction.

Skip on Keyboard Flag (KSF)

Octal Code: 6031
Operation: Increments the contents of the PC if the keyboard flag is set, so that the next instruction is skipped.

Clear Keyboard Flag (KCC)

Octal Code: 6032
Operation: Clears the keyboard flag and AC and sets the reader run flip-flop. This action allows the hardware to begin assembling the next input character in the TTI register. If the reader is activated and there is tape in the reader, a serial character is read from tape and is assembled in the TTI register. The keyboard can also load characters

into the TTI register provided the reader is deactivated. In either case, the keyboard flag is set when the character is assembled on the TTI.

Read Keyboard Buffer Static (KRS)

Octal Code: 6034
Operation: ORs the contents of TTI register with AC4 through 11, and leaves the result in AC4-11. This is termed a static command because neither the AC nor keyboard flag is cleared.

Set/Clear Interrupt Enable (KIE)

Octal Code: 6035
Operation: Sets or clears the interrupt enable flip-flop as defined by AC11. If AC11 is asserted, generates an interrupt request for a keyboard or teleprinter flag. If AC11 is negated, interrupt requests cannot be generated.

Read Keyboard Buffer Dynamic (KRB)

Octal Code: 6036
Operation: Performs the combined operations of the KCC and KRS instructions. Clears the AC and keyboard flag and transfers the contents of the TTI register to AC4 through AC11. This instruction also sets the reader run flip-flop to begin assembly of another character in the TTI register. When this operation is complete, the keyboard flag is set to indicate another character is available.

A typical TTI instruction sequence for keyboard (manual) input is:

```
LOOK,   KSF          /SKIP IF KEYBOARD FLAGS
        JMP  LOOK    /JUMP BACK & TEST FLAG AGAIN
        KRB          /TRANSFER TTI CONTENTS INTO AC
```

This sequence waits for the TTI to set its flag, indicating that it has a character ready to be transferred. It then skips to the KRB command which causes the character to be transferred from the TTI to the AC.

The computer clears all flags which are on the clear flag bus (including both the keyboard flag and the reader run enable) when the console CLEAR pushbutton is depressed. This means that the user program must set the reader run enable to obtain data from the reader. The instruction sequence given below is a typical TTI instruction sequence for both keyboard and reader input.

If this sequence of instructions is made a subroutine of the main program, it can be accessed each time an input character is desired. Consequently,

Load Teleprinter and Print (TPC)

Octal Code: 6044
Operation: Transfers AC bits 4-11 to the TTO and starts shifting the character out to the printer/punch unit. This instruction does not clear the teleprinter flag. This instruction can be micro-programmed with TCF to produce TLS.

Skip On Printer or Keyboard Flag (TSK)

Octal Code: 6045
Operation: Skips the next instruction if the printer or keyboard flag is set and the interrupt enable flip-flop is set.

Load Teleprinter Sequence (TLS)

Octal Code: 6046
Operation: This instruction combines TCF and TPC. The teleprinter flag is cleared and the contents of AC bits 4-11 are transferred to the TTO, where the hardware shifts the character out to the printer/punch unit. When the TTO has finished outputting the character and is ready for another character, the teleprinter flag is set. The whole operation, from the time at which the TLS has cleared the flag and TTO starts character transfer, until the time the hardware finishes with the character and again sets the flag, requires 100 ms at 110 baud.

A typical TTO instruction sequence is:

```
        .
        .
        .
        CLA
        TAD X       /PUT CHARACTER CODE INTO AC FROM
                    /LOCATION X
        TLS         /LOAD TTO FROM AC & PRINT/PUNCH
FREE,   TSF         /TEST FLAG TO SEE IF DONE PRINTING,
                    /SKIP IF = 1
        JMP FREE    /TEST FLAG AGAIN
        CLA         /CLEAR CHARACTER CODE FROM AC
        .
        .
        .
```

This sequence sends one character to the TTO and waits for printing/punching before sending another character. It does not require that the flag be set to output the character. By making the instruction sequence a subroutine of a larger program, it can be accessed by a JMS each time a character is to be output. Assume that the subroutine is entered with the character code in the AC:

```
TYPE,   0
        TLS         /LOAD TTO FROM AC AND PRINT/PUNCH
        TSF         /TEST FLAG, SKIP IF = 1
        JMP .-1
        .
        .
```

```
        KCC             /CLEAR TTI FLAG, SET READER RUN, CLEAR
                        /AC
        .
        .
        .
READ,   0               /STORE PC HERE FOR RETURN ADDRESS
        KSF             /SKIP IF FLAG = 1
        JMP .-1         /TEST FLAG AGAIN
        KRB             /READ CHAR INTO AC
        JMP I READ      /EXIT TO MAIN PROGRAM
        .
        .
        .
```

Teleprinter/Punch

On program command a character is transferred from the AC to the output shift register (TTO) for transmission to the teleprinter/punch unit. The Teletype control generates the start space, shifts the eight character bits serially into the printer selector magnet of the Teletype unit, and then generates the stop marks. Bit transfer from the TTO to the teleprinter punch unit is at the normal Teletype rate. A character transfer requires 100 ms for completion at 110 baud. The teleprinter flag is set when the last bit of the character code is sent to the teleprinter/punch, indicating that the TTO is ready to receive a new character from the AC. The flag activates the program interrupt synchronization element and the instruction skip element. When using instruction skip, the program checks the flag by means of the TSF instruction. If the flag is set, the program issues the TLS instruction, which clears the flag and sends a new character from the AC to the TTO. AC-to-TTO transfer time is short compared to the print/punch time, so the program must wait for the flag to set before issuing another TLS.

Programming

Instructions for use in outputting teletype data are as follows:

Set Teleprinter Flag (TFL)

Octal Code: 6040
Operation: Sets the teleprinter flag to ready the logic for another character.

Skip on Teleprinter Flag (TSF)

Octal Code: 6041
Operation: If the teleprinter flag is set, increments the contents of the PC by one so that the next sequential instruction is skipped.

Clear Teleprinter Flag (TCF)

Octal Code: 6042
Operation: Clears the teleprinter flag. This instruction can be microprogrammed with TPC.

By rearranging this subroutine, the 100 ms (at 110 baud) spent waiting for the character to be output and the flag to be set is used to continue the main program, making more efficient use of program time.

```
TYPE,   0           /TEST FLAG TO SEE IF TELEPRINTER FREE,
        TSF         /SKIP IF YES OR . . .
        JMP .—1     /WAIT TILL IT IS BY TESTING AGAIN AND
                    /AGAIN
        TLS         /OUTPUT CHARACTER
        CLA         /CLEAR CHARACTER FROM AC
        JMP I TYPE  /EXIT TO CONTINUE PROGRAM
```

This subroutine tests the flag first, and waits only if a previous character is still being outputted. It clears the AC, exits immediately after sending the character to the TTO, and continues to run the user's program, instead of waiting while the Teletype (a much slower I/O device) is typing/punching the preceding character. The computer clears all flags which are on the clear flag bus (including teleprinter flags) when the console CLEAR pushbutton is depressed. This means that the user program must account for setting the teleprinter flag initially and after each TCF (if any), or the program hangs up in the wait loop of the print routine. The only way to set the flag is by initializing it. This instruction should appear among the first few executed, and must appear before any attempt to output a character. The following example initializes the flag as the first instruction of the program and makes optimum use of the punch/print time.

```
BEGIN,  TFL
        .
        .
        .
TYPE,   0
        TSF          /SKIP IF FLAG = 1 OR . . .
        JMP .—1      /WAIT UNTIL IT IS LOAD TTO &
        TLS          /TYPE CHARACTER
        CLA          /CLEAR CHARACTER FROM AC
        JMP I TYPE   /EXIT & CONTINUE PROGRAM WHILE
        .            /TELEPRINTER IS FINISHING CHARACTER
        .
        .
```

PAPER TAPE READER AND PUNCH OPTIONS

The options available for paper tape facilities are listed below.

PR8-E	Reader (with Control Unit)
PP8-E	Punch (with Control Unit)
PC8-E	Reader/Punch (with Control Unit)
PR8-ES	Reader (with Control Unit)

Type PR8-E Paper Tape Reader

The PR8-E reader senses eight-hole unoiled grey perforated paper tape photoelectrically at a maximum rate of 300 characters per second. The control unit of the PR8-E plugs into the OMNIBUS and controls the ac-

tion of the reader from program instructions. All connections between the control unit and the reader are made using H854 connectors.

A read operation is initiated by an RFC instruction from the computer. The control unit, in turn, initiates tape movement and sensing of a character, transfers the character to its reader buffer (RB), and sets its device flag to indicate that a character is available for transfer to the computer. The computer senses the reader flag by issuing an RSF instruction, and transfers the character from the RB to AC04 through 11 by issuing an RRB instruction. The RRB instruction also clears the reader flag to ready the unit for another read operation.

The control unit also contains an interrupt enable flip-flop. This flip-flop, controlled by program instructions, determines whether the reader can generate an interrupt request to the program interrupt facility. When set by an RPE instruction or initialize input, this flip-flop enables generation of an interrupt request from the reader flag being set. When cleared by a PCE instruction, this flip-flop inhibits interrupt requests.

Programming
Instructions for operating the reader are as follows:

Set Reader/Punch Interrupt Enable (RPE)

Octal Code: 6010
Operation: Sets the reader/punch interrupt enable flip-flop so that an interrupt request can be generated when reader or punch flag is set.

Skip on Reader Flag (RSF)

Octal Code: 6011
Operation: Senses the reader flag; if it contains a binary one, increments the PC by one so that the next sequential instruction is skipped.

Read Reader Buffer (RRB)

Octal Code: 6012
Operation: ORs the content of the reader buffer into AC4-11 and clears the reader flag. This command does not clear the AC.

Reader Fetch Character (RFC)

Octal Code: 6014
Operation: Clears the reader flag, loads one character into the RB from the tape, and sets the reader flag when the RB is full.

Read Buffer and Fetch New Character (RRB, RFC)

Octal Code: 6016
Operation: Combines RRB and RFC. The contents of the reader buffer is ORed into the AC. The flag is immediately cleared, and a new character is read from tape into the reader buffer. The flag is then set.

Clear Reader/Punch Interrupt Enable (PCE)

Octal Code: 6020
Operation: Clears the reader/punch interrupt enable flip-flop so that interrupt requests cannot be generated.

A program sequence loop to read a character from perforated tape can be written as follows:

```
        RFC         /FETCH CHARACTER FROM TAPE
LOOK,   RSF         /SKIP IF READER FLAG = 1
        JMP LOOK    /JUMP BACK & TEST FLAG AGAIN
        CLA         /CLEAR AC
        RRB         /LOAD AC FROM RB, CLEAR READER FLAG
```

PP8-E Paper Tape Punch

The PP8-E paper tape punch consists of a control unit and a punch assembly that perforates eight-hole unoiled grey perforated paper tape at a rate of 50 characters per second. The control unit plugs into the OMNIBUS of the PDP-8/E, and controls the operation of the punch from program instructions. All connections between the control unit and the punch are made using H854 connectors.

A punch operation can be performed using a PCF instruction, followed by a PPC instruction or by a PLS instruction. In either event, the punch flag and punch buffer (PB) are cleared. An eight-bit character is then loaded into the PB from AC04 through 11 and the character is perforated on tape. After the character is punched, the punch flag is set to denote that the punching operation is complete. The punch flag is sensed by a PSF instruction to begin another punch operation.

The control unit also contains an interrupt enable flip-flop that is set or cleared by program instruction. When set by an RPE instruction or INITIALIZE, this flip-flop enables gating of an interrupt request to the program interrupt facility when the reader or punch flag is set. When cleared by a PCE instruction, this flip-flop prevents interrupt requests.

Programming

Instructions for the punch are as follows:

Set Reader/Punch Interrupt Enable (RPE)

Octal Code: 6010
Operation: Sets the reader/punch interrupt enable flip-flop so that an interrupt request can be generated when punch or reader flag is set.

Clear Reader/Punch Interrupt Enable (PCE)

Octal Code: 6020
Operation: Clears the reader/punch enable flip-flop so that interrupt requests cannot be generated.

Skip on Punch Flag (PSF)

Octal Code: 6021
Operation: Senses the punch flag; if it contains a binary one, increments the PC by one so that the next sequential instruction is skipped.

Clear Punch Flag (PCF)

Octal Code: 6022
Operation: Clears the punch flag in preparation for receiving a new character from the computer.

Load Punch Buffer and Punch Character (PPC)

Octal Code: 6024
Operation: Transfers the eight-bit character in AC4-11 into the PB, then punches that character. The instruction does not clear the punch flag or the PB.

Load Punch Buffer Sequence (PLS)

Octal Code: 6026
Operation: Clears the punch flag, transfers the contents of AC4-11 into the punch buffer, punches the character in the PB on tape, and sets the punch flag when the operation is completed.

A program sequence loop to punch a character when the punch buffer is free can be written as follows:

```
FREE,   PSF       /SKIP IF PUNCH FLAG = 1
        JMP FREE  /JUMP BACK & TEST FLAG AGAIN
        PLS       /CLEAR PUNCH FLAG & PB, LOAD PB
                  /FROM AC, PUNCH CHARACTER, SET
                  /PUNCH FLAG WHEN DONE
```

PC8-E Reader/Punch

The PC8-E consists of a reader and punch mounted on the same chassis and a control unit which plugs into the OMNIBUS and controls the action of the reader/punch from program instructions. All connections between the control unit and reader/punch are made using H854 connectors.

The reader portion of the PC8-E operates in the same manner as the PR8-E. Similarly, the punch portion of the PC8-E operates in the same manner as the PP8-E.

Specifications

Tape Type	1-inch fan-folded unoiled grey paper
Channels	8 data channels plus feedhole
Read Character Rate (Continuous)	300 characters/second
Read Character Rate (Start-Stop Mode)	25 characters/second
Punch Character Rate	50 characters/second

Programming
The PC8-E uses the same instructions (and programming sequence) as the PR8-E and PP8-E.

Type PR8-ES Paper Tape Reader
The PR8-ES reader senses eight-hole oiled or unoiled perforated paper tape photoelectrically at a maximum rate of 110 characters per second. The control unit of the PR8-ES plugs into the PDP-8/E OMNIBUS, and controls the action of the reader from program instructions. The reader requires approximately one square foot of table space. An On Line/Off Line switch controls the reader lamp.

Programming
Programming for the PR8-ES is identical to that for the PR8-E reader.

CRT DISPLAYS
Type VC8-E Oscilloscope Display Control
The VC8-E oscilloscope display option permits the PDP-8/E user to plot points on suitable CRT display oscilloscopes. It is a two-axis digital-to-analog converter and intensifying circuit that provides deflection and intensity signals, which are then applied to the input amplifier circuitry of such display units as the Type VR03A or the Type VR14 oscilloscopes. The control circuit for the VC8-E is located on a PDP-8/E module (M869), which plugs into the OMNIBUS. It can be connected to and used with the Type VA8-E Character Generator and the Type 374A Light Pen.
Display time may be optimized by using program control to determine whether the mode of operation will be fast or slow; fast for continuous plots, slow for random point plotting. This feature is especially useful in conjunction with an electromagnetic type of oscilloscope, such as the VR14.

The basic system of the VC8-E consists of the following circuitry:

a. OMNIBUS interface, IOT decoding, skip, clear AC, and interrupt control.
b. X-axis buffer-counter, D/A converter, filter and summing amplifier, and bipolar line driver.
c. Y-axis buffer-counter, D/A converter, filter and summing amplifier, and bipolar line driver.
d. Z-axis control, which consists of provision for intensity signal necessary for the VR03A oscilloscope and intensity and channel select signals necessary for the VR14 oscilloscope.
e. Light pen receiver circuitry and control.
f. Provision for inclusion of character generator, Type VA8-E.
g. Generation of voltage for analog circuitry.

The VC8-E interfaces with the VR03A or the VR14 by means of two different connector assemblies. Use of various jumpers permits the same control board to control any of several oscilloscopes.
Coordinate data is loaded into the D/A converters for the X and Y registers from AC02-11 in the form of ten-bit characters. Unipolar D/A output is converted to bipolar output by the summing amplifier for each axis, providing the deflection and intensifying pulses needed to plot data on the oscilloscopes.

When a point has been displayed and intensified, the Point Done Flag is set. When a character has been displayed, the Character Done Flag is set. When a light pen hit occurs, the Light Pen Flag is set. Each of the flags causes an interrupt, and each can be tested with a SKIP IOT. If the flag is set, a skip will be produced and the flag will be reset. If a flag is not set, no skip will be produced.

Programming

The instructions for outputting data to the oscilloscope display are as follows:

Load X Register from AC (DLXA)

Octal Code: 6060
Operation: Loads the X register with new x-axis data from AC2-11 by jam transfer.

Load X Register from AC, Clear AC (DLXB)

Octal Code: 6061
Operation: Loads the X register with new x-axis data from AC2-11 and clears the AC.

Load X Register from AC and Intensify (DLXC)

Octal Code: 6062
Operation: Loads the X register with new x-axis data from AC2-11 and intensifies the oscilloscope.
Display Time: See NOTE below.

Load X Register from AC, Clear AC, Intensify (DLXD)

Octal Code: 6063
Operation: Loads the X register with new x-axis data from AC2-11, clears the AC, and intensifies the oscilloscope.
Display Time: See NOTE below.

Set Small Character Size (DSIA)

Octal Code: 6064
Operation: Sets small size of characters obtained from the character generator, Type VA8-E.

Set Large Character Size (DSIB)

Octal Code: 6065
Operation: Sets large size of characters obtained from the character generator, Type VA8-E.

Increment X and Intensify (DINX)

Octal Code: 6066
Operation: Increments the X register and intensifies the oscilloscope.
Display Time: See NOTE below.

Display Character from AC (DSC)

Octal Code: 6067
Operation: Displays the six-bit character received from the character generator, Type VA8-E, via AC6-11.

Load Y Register from AC (DLYA)

Octal Code: 6070
Operation: Loads the Y register with new y-axis data from AC2-11.

Load Y Register from AC and Clear AC (DLYB)

Octal Code: 6071
Operation: Loads the Y register with new y-axis data from AC2-11 and then clears the AC.

Load Y Register from AC and Intensify (DLYC)

Octal Code: 6072
Operation: Loads the Y register with new y-axis data from AC2-11 and intensifies the oscilloscope.
Display Time: See NOTE below.

Load Y Register from AC, Clear AC, Intensify (DLYD)

Octal Code: 6073
Operation: Loads the Y register with new y-axis data from AC2-11, clears the AC, and intensifies the oscilloscope.
Display Time: See NOTE below.

Set VR14 Channel 1 (DCHI)

Octal Code: 6074
Operation: Selects Channel 1 of the VR14 oscilloscope.

Set VR14 Channel 2 (DCH2)

Octal Code: 6075
Operation: Selects Channel 2 of the VR14 oscilloscope.

Increment Y and Intensify (DINY)

Octal Code: 6076
Operation: Increments the Y register and intensifies the oscilloscope.
Display Time: See NOTE below.

Clear Status, Interrupt Enable Registers (DCSI)

Octal Code: 6077
Operation: Clears the status and interrupt enable registers.

Load Interrupt Enable Register from AC (DIEN)

Octal Code: 6050
Operation: Loads the content of AC9-11 into the interrupt enable register.

Skip on Point Done Flag and Clear Point Done Flag (DSPD)

Octal Code: 6051
Operation: Senses the Point Done Flag; if it is set, increments the content of the PC by one so that the next sequential instruction is skipped, and then clears the Point Done Flag.

Skip on Character Done Flag, Clear Character Done Flag (DSCD)

Octal Code: 6052
Operation: Senses the character done flag; if it is set, increments the content of the PC by one so that the next sequential instruction is skipped, and then clears the character done flag.

Skip on Light Pen Flag, Clear Light Pen Flag (DSLP)

Octal Code: 6053
Operation: Senses the light pen flag; if it is set, increments the content of the PC by one so that the next sequential instruction is skipped, and then clears the light pen flag.

Skip on Any Display Flag (DSDF)

Octal Code: 6054
Operation: Senses the point done flag, the character done flag, and the light pen flag; if any of these is set, increments the content of the PC by one so that the next sequential instruction is skipped, and then clears the flag which was set.

Read Interrupt Flags into AC, Clear Flags (DRIS)

Octal Code: 6055
Operation: ORs the status of the interrupt flags into AC9-11 and clears the corresponding flags. This also permits setting a flag while it is being read. The status of the Light Pen Flag is transferred to AC9, the Character Done Flag to AC10, and the Point Done Flag to AC11.

Set Fast Plotting Mode (DSFF)

Octal Code: 6056
Operation: Establishes that mode of operation will be fast, for continuous plotting, by setting the mode flip-flop.

Clear Fast Plotting Mode (DCFF)

Octal Code: 6057
Operation: Clears the mode flip-flop, establishing that mode of operation will be slow, for random point plotting.

NOTE on Display Times

The display times of those instructions that include intensification depend upon the type of oscilloscope used and the mode of operation selected:

	SLOW MODE	FAST MODE
VR14	24.95 μs	4.25 μs
VR03A	2.45 μs	1.8 μs

Type VR03A Oscilloscope Display

The VR03A consists of a Tektronix Model RM602 oscilloscope display and mounting hardware (laboratory peripheral panel, Type H945-AA). The RM602 is a solid state, five-inch, 1-MHz display with BNC inputs used with the VC8-E point plot display control. It provides accurate display of information from X, Y, and Z signal inputs. Interface with the VC8-E is by means of connector assembly BC01L-10 (10 feet), BC01L-25 (25 feet), or BC01L-50 (50 feet), with ten feet the standard length.

Type VR14 Oscilloscope Display

The VR14 is a compact solid-state CRT display with self-contained power supplies and a viewing area of 6¾ in. x 9 in. The VR14 can plot 2000 random points and up to 75 in. of vector with no flicker. X/Y deflection speed is 600 ns intensified, 400 ns nonintensified, and less than 25 μs is required for a maximum deflection step in any direction. The VR14 has provision for differential inputs, and is recommended over the VR03A wherever possible because of ground separation (analog and digital). Interface with the VC8-E is by means of connector assembly BC01K-10 (10 feet), BC01K-25 (25 feet), or BC01K-50 (50 feet), with ten feet the standard length.

Type 374A Light Pen

The 374A is a photoelectric light pen that detects the presence of information displayed on a CRT. If the pen is held against the face of the CRT at a bright point, the light pen display flag is set, and this activates the computer instruction skip facility.

Type VA8-E Character Generator

The VA8-E is an optional adjunct to the VC8-E that consists of a PDP-8/E module which plugs into the OMNIBUS. It is a 65-μs, 64-character, ASCII character generator with a read-only, MSI memory. The ASCII characters corresponding to codes 240 through 337 are available. See Appendix C.

Type VS8-E Display Control for VT01 Storage Tube Display

The VS8-E Display Control utilizes the VT01 storage tube display with a 6½ inch by 8¼ inch screen size. The H306 joystick graphic input device is an optional adjunct.

The controller portion of the graphic system is an analog stroke vector generator that provides programmed selection and control of the VT01 storage tube display. Hardware capabilities include: single-command point plotting, linear vectors, circular vectors, and arc vectors. Alphanumeric character generation is accomplished by the use of an efficient, programmable subroutine supplied with the system.

The controller accepts analog input signals produced by the H306 joystick or other compatible source on the X or Y cursor channels. The system provides a means to display these signals as a visible but non-stored cursor on the VT01, and to perform the required analog-to-digital conversion to enter these signals into the computer.

Hardware Features

a. No refresh is necessary.
b. Drift and flicker are eliminated.
c. Linear stroke vector generation at 200 full-screen vectors/second, or 1600 short vectors/second.
d. Circle generation accomplished by single-command circular stroke generator.
e. Point-plotting capable of addressing a page-size of 32 x 32 cm with 1024 x 1024 points.
f. Joystick-controlled pointer incorporating a write-through cursor.
g. Origin shifting and scaling (zooming) to view area 21 x 16.3 cm.

Programming

The following instructions are used for the VS8-E Display Control:

Skip if Cursor Interrupt Not Set (SNC)

Octal Code: 6051
Operation: Senses the condition of the Cursor Interrupt Flag which produces an interrupt request when set by operation of the interrupt button on the joystick. This flag and all other flags of the VS8-E are initially cleared when the computer is started.

Clear Interrupt Flag (CFF)

Octal Code: 6052
Operation: Clears the Cursor Interrupt Flag.

Select Analog Comparator (SAC)

Octal Code: 6062
Operation: Sets the Analog Comparator to compare the analog content of any one of six analog sources with the content of the digital-to-analog converter. The analog sources are chosen according to a three-bit binary code, which establishes the parameter for choosing the wanted register according to the content of AC2, AC3, and AC6.

AC BIT 2 3 6	FUNCTION
0 0 0	READ X INTEGRATOR
0 0 1	READ Y INTEGRATOR
1 0 0	READ X SAMPLE/HOLD
1 0 1	READ Y SAMPLE/HOLD
1 1 0	DISPLAY CURSOR AND READ X CURSOR SIGNAL
1 1 1	DISPLAY CURSOR AND READ Y CURSOR SIGNAL

Load and Select Format (LDF)

Octal Code: 6063
Operation: Establishes the mode in which a wanted graphic is to be produced according to a two-bit binary code. This code

determines whether the desired vector will be linear absolute or relative, whether the point plot mode will be used, and if the cursor will be displayed. This code establishes the parameters for these formats by loading the content of AC2 and AC3 into the Format Register. The LDF instruction must precede the LDX and LDY instructions.

AC BITS 2 3	FORMAT
0 0	VECTOR MODE, RELATIVE INPUT
0 1	VECTOR MODE, ABSOLUTE INPUT
1 0	POINT MODE
1 1	DISPLAY CURSOR

NOTE

SAC and LDF do not affect flag status.

Load AC into X Sample and Hold (LDX)

Octal Code: 6064

Operation: Loads the X-axis sample and hold register with the contents of AC2-11. The LDX instruction immediately clears the Ready Flag and sets the Ready Flag after 100 μs + or — 20 μs.

Load AC into Y Sample and Hold (LDY)

Octal Code: 6065

Operation: Loads the Y-axis sample and hold register with the contents of AC2-11. LDY instruction immediately clears the Ready Flag and sets the Ready Flag after 100 μs + or — 20 μs.

Execute Microprogrammed Instruction (EXC)

Octal Code: 6066

Operation: Executes the desired vector in the Function Register and AC6-11 (bit 4 inhibits bit 6). The parameter word establishes long or short formats, circular vectors, display erasure, reset of the integrators, and intensification of the vector. The EXC instruction clears an existing Ready Flag and sets the Ready Flag as follows:

a. After 20 us + or — 5us for a point or vector continue
b. After 250 us for short vectors
c. After 4.05 ms for long vectors
d. After 500 ms for an erase.

BIT	FUNCTION WHEN SET	WHEN NOT SET
2	See under 6063	
3	See under 6063	
4	Set execution to short vector speed.	Normal stored mode with speed set by bit 6

BIT	FUNCTION WHEN SET	WHEN NOT SET
5	Not used	
6	Execute long vector unless inhibited by bit 4.	Execute short vector
7	Set vector continue when flag is raised.	Vector continue disabled. Can also be disabled using 6062 or 6063.
8	Execute circle vector when enabled by bit 2.	Execute linear vector when enabled by bit 2.
9	Erase the scope.	No action
10	Reset integrators and sample/holds	No effect
11	Intensify vector	Blank vector

Execute Long Vector Microprogrammed Instruction (EXL)

Octal Code: 6067
Operation: Execute long vector with bit positions in the same manner as EXC (6066). Long vector is inhibited by bit 4. (Operation requires 4.05 ms.)

Skip if Ready Flag is Set (SRF)

Octal Code: 6071
Operation: Determines when the controller is ready to perform the next execute instruction. The Ready Flag produces an interrupt when set. The Ready Flag can be set by pressing the erase pushbutton on the VT01 unit. Normally, however, the state of this flag is determined by the controller. This flag is initially cleared when the computer is started and by an LDX, LDY, or EXC instruction.

Clear Ready Flag (CRF)

Octal Code: 6072
Operation: Clear the Ready Flag.

Skip if D/A Greater than Analog Comparison Signal (SDA)

Octal Code: 6073
Operation: Used to determine the digital equivalent of the selected analog holding register. This instruction is used with the SAC (6062) instruction. When the voltage in the analog comparator is greater than the selected value (the value in the register), the next instruction is skipped.

Load D/A (LDA)

Octal Code: 6074
Operation: Loads the content of AC2-11 into the analog comparator, and is used with SDA (6073) to determine the digital value of the content of the selected analog holding register. This instruction does not change flag states.

Type VT01 Storage Tube Display
The VT01 is a modified Tektronix Model 611 direct-view storage tube with a resolution of 300 stored line pairs vertically and 300 stored line pairs horizontally. Dot writing time is 20 us, with a full screen erase time of 500 ms. The VT01 can display 4,000 clearly legible flicker-free characters or 30,000 discrete resolvable points and is capable of remote control erasing and write-through.

Type H306 Joy Stick Cursor Controller
The H306 is a self-contained unit that connects to the VT01 by a 12-wire multiconductor shielded cable. The H306 is a controlled pointer, incorporating a write-through cursor.

X/Y PLOTTER OPTIONS
Type XY8/E Incremental Plotter Control
The XY8/E Incremental plotter control provides the control logic and interface necessary to operate an encoded or unencoded digital incremental plotter. It operates with a variety of plotters to display data graphically on paper or film.

Except for setting the coordinates at which plotting begins, all plotter operations are controlled by the plotter control logic and the processor. A series of functions controlled by IOT instructions initializes the plotter control logic, generates program interrupt requests to indicate change in status, and initiates a plotting operation.

The principles of operation are basically the same for all plotters. Drum plotter operations are described below:

Bidirectional rotary stepping motors are employed for both the X and Y axes. Recording is produced by movement of a pen in relation to the surface of graph paper, with each instruction resulting in an incremental step. X-axis deflection is derived from the motion of the drum; Y-axis deflection is derived from the motion of the pen carriage. Further instructions lower and raise the pen to and from the surface of the paper. Inputs to the plotter from the digital signal source consist of drum up, drum down, carriage right, carriage left, pen up, and pen down pulses. All recording (discrete points, continuous curves, or symbols) is accomplished by the incremental stepping action of the paper drum and pen carriage.

Controls on the plotters permit single-step or continuous-step manual operation of the drum and carriage, and manual control of the pen solenoid. The recorder and control are connected to the computer program interrupt and instruction skip facility.

The entire interface is contained on a PDP-8/E module which plugs into the OMNIBUS.

NOTE

Unencoded Plotters can be:
CAL-COMP 500 Series
CAL-COMP 600 Series
HOUSTON INSTRUMENTS DP-1

Encoded Plotters can be:
CAL-COMP 600 Series
CAL-COMP 700 Series
CAL-COMP 800 SERIES (MICROFILM RECORDERS)

Programming

The following IOT instructions are used to operate the digital incremental plotters:

Clear Interrupt Enable (PLCE)

Octal Code: 6500
Operation: Clears the interrupt enable flip-flop.

Skip Plotter Flag (PLSF)

Octal Code: 6501
Operation: Senses the Plotter Flag, and, if it is set, increments the contents of the PC by one so that the next sequential instruction is skipped.

Clear Plotter Flag (PLCF)

Octal Code: 6502
Operation: Clears the Plotter Flag preparatory to issuing a plotter operation command.

Pen Up (PLPU)

Octal Code: 6503
Operation: Raises the plotter pen from the surface of the graph paper (unencoded plotters only).

Load Direction Register, Set Flag (PLLR)

Octal Code: 6504
Operation: Loads the direction register from AC6-11, which performs the following function:

Unencoded Plotter

AC BIT	FUNCTION
6	Pen Right
7	Pen Left
8	Drum Down
9	Drum Up

Encoded Plotter Function

AC05-11	PAPER PLOTTERS	FILM PLOTTERS
10	+y	+y
11	+x +y	+x +y
12	+x	+x
13	+x −y	+x −y

14	−y	−y
15	−x −y	−x −y
16	−x	−x
17	−x +y	−x +y
30	not used	CRT Shift
31	Pen Up	Beam Off
32	Pen Down	Beam On
33	Start Zip	not used
34	Block Code	−z (+ Aux 1)
35	Plot Code	+z (+ Aux 2)
36	Start Incr.	not used
37	Sync	Sync
50	+y/2	
51	+x/2 +y/2	
52	+x/2	
53	+x/2 −y/2	
54	−y/2	
55	−x/2 −y/2	
56	−x/2	
57	−x/2 +y/2	
70	+x +y/2	
71	−x +y/2	
72	+x/2 +y	
73	−x/2 +y	
74	+x −y/2	
75	−x −y/2	
76	+x/2 −y	
77	−x/2 −y	

Pen Down (PLPD)

Octal Code: 6505
Operation: Lowers the pen to the surface of the graph paper (unencoded plotters only).

Clear Flag, Load Direction Register, Set Flag (PLCF, PLLR)

Octal code: 6506
Operation: This microinstruction combines octal instructions 6502 and 6504. It clears the Plotter Flag, loads the direction register from AC6-11, then sets the flag.

Set Interrupt Enable (PLSE)

Octal code: 6507
Operation: Sets the interrupt enable flip-flop.
Any sequence of programmed IOTs must assume that the pen location is known at the start of the routine, as there is no way to specify an absolute location in an incremental plotter except by the manual controls on the recorder. During a subroutine, the PDP-8/E can track the location of the pen on the paper by counting the instructions that increment the position of the pen and the drum.

Type XY8-EA Digital Incremental Plotter

The XY8-EA consists of the XY8-E interface described above and the Calcomp (California Computer Products) Model 565 Digital Incremental Plotter. The Model 565 is a high-speed, drum-type plotter, capable of performing up to 18,000 steps per minute. Each of these steps causes the drum or pen carriage to move a fixed increment in either a positive or negative direction. The size of this increment can be 0.01 inch, 0.003 inch, or 0.1 mm, depending on the gear ratios used for the drum and carriage drives.

A bidirectional roll paper feed and takeup mechanism accepts chart paper rolls 12 inches wide by 120 feet long. The paper is driven by pins on the drum which engage holes on both edges of the paper to maintain registration between the recording pen and the paper. If desired, single sheets of chart paper may be used for plotting instead of the roll paper.

Type XY8-EB Digital Incremental Plotter

The XY8-EB consists of the XY8-E interface together with the Calcomp Model 563 Digital Incremental Plotter. This is very similar to the Type XY8-EA, except that the Model 563 accepts a paper width of 30 inches.

LINE PRINTER OPTION

LE8 Line Printer

The LE-8 line printer offers the user a low-cost, high-speed, flexible method of printing computer output at a rate dependent upon the option selected. It accepts ASCII characters from the AC.

Each character is selected from the set of 64 (or 96) available by means of six-bit or seven-bit binary code. (Appendix E lists the ASCII code for each character.) Each code is loaded separately from the computer into a 20-character (or, in the case of the 132-column model, 22-character) core storage Line Printer Buffer from AC 6-11 (or AC 5-11), with the least significant bit appearing in AC11. After each code is transferred into the Line Printer Buffer, the Line Printer Done Flag appears, indicating that the printer is ready to receive the next character. When the Line Printer Buffer is filled, or a control character has been received, the print cycle is initiated. Character codes not in the character set in Appendix E are printed as spaces. The line feed command and carriage return command are similar to the corresponding commands in the Teletype. The form feed command advances the paper to the top of the page. The Printer Done Flag is set after each of these operations.

During the print cycle, the paper and inked ribbon pass between a row of 80 hammers (132 in the 132-column model) and the continuously rotating drum that contains all of the available characters. Variable reluctance pickoffs scan the stored characters in synchronism with the rotating characters, and the control system actuates the appropriate hammer as the desired character approaches the print position. The full line is printed in 20-column segments, with one drum revolution required for each segment. After the last character of a line is printed, the Line Printer Buffer is cleared automatically.

There are no operator controls in the control module. The following controls are on the printer:

TOP OF FORM—Advances paper to top-of-form position; disabled in on-line mode

PAPER STEP—Advances paper one line; disabled in on-line mode

ON LINE/OFF LINE—Selects mode of operation for the printer

MASTER CLEAR—Initializes printer to ensure proper state of electronic elements

PRINT INHIBIT—Inhibits print hammers.

The line printer is available in any of the following combinations:

LE8-FA	80 columns	64 characters	60 Hz
LE8-FB	80 columns	64 characters	50 Hz
LE8-HA	80 columns	96 characters	60 Hz
LE8-HB	80 columns	96 characters	50 Hz
LE8-JA	132 columns	64 characters	60 Hz
LE8-JB	132 columns	64 characters	50 Hz
LE8-KA	132 columns	96 characters	60 Hz
LE8-KB	132 columns	96 characters	50 Hz

The interface is contained on one PDP-8/E module, which plugs into the OMNIBUS.

The specifications for the LE8 line printer are as follows:

Printable characters	
character set	64 or 96
type	Open Gothic print
size	Typically 0.095 inches high and 0.065 inches wide
code format	ASCII
characters per line	80 or 132
drum speed	1760 rpm (64 character drum)
Print rate	
80 column model	
64 character	356 Lines/minute, columns 1-80 460 Lines/minute, columns 1-60 650 Lines/minute, columns 1-40 1110 Lines/minute, columns 1-20
96 character	253 Lines/minute, columns 1-80 330 Lines/minute, columns 1-60 478 Lines/minute, columns 1-40 843 Lines/minute, columns 1-20
132 column model	
64 character	245 Lines/minute, columns 1-132 290 Lines/minute, columns 1-110 356 Lines/minute, columns 1-88 460 Lines/minute, columns 1-66 650 Lines/minute, columns 1-44 1110 Lines/minute, columns 1-22

96 character	173 Lines/minute, columns 1-132 205 Lines/minute, columns 1-110 253 Lines/minute, columns 1-88 330 Lines/minute, columns 1-66 478 Lines/minute, columns 1-44 843 Lines/minute, columns 1-22
Format	Top-of-form control, single line advance and perforation step over.
Paper feed	One pair of pin-feed tractors for ½-inch hole center, edge-punched paper. Adjustable for any paper width from 4 inches to 9-⅞ inches on the 80-column model; or a maximum width of 14-⅞ inches for the 132 column model.
Paper slew speed	13 inches per second
Print area	8 or 13.2 inches wide, left justified
Character Spacing	10 characters per inch
Line spacing	6 lines per inch for 80-column, 6 or 8 lines for 132-column printer
Line advance time	20 milliseconds
Character synchronization	Variable reluctance pick-offs sense drum position

Printer Dimensions

	80 column	132 column
Height	46 inches	46 inches
Width	24 inches	48 inches
Depth	22 inches	25-inches
Weight	200 pounds	220 pounds

Printer Power Requirements

115 vac + or − 10%, 60 Hz + or − 3 Hz, single phase, 300 watts
or
240 vac + or − 10% 50 Hz + or − 3 Hz, single phase, 300 watts

Signal cable	25 foot interconnecting signal cable is supplied with system
Paper	
Type	standard fanfold, edge punched
Dimensions	4 inches to 9-⅞ inches wide (80 column) 4 inches to 14-⅞ inches wide (132 column) with 11 inches between folds

weight	
(single copy)	15 pound bond (minimum)
multi copy)	12 pound bond with single-shot carbon for up to six parts
Ribbon	
type	inked roll
width	9 inches (80 column); 14 inches (132 column)

Programming

The IOT instructions which command the line printer are:

Skip on Character Flag (PSKF)

Octal Code: 6661
Operation: Senses the content of the line printer done flag; if it contains a binary 1, the contents of the PC are incremented by one so that the next sequential instruction is skipped.

Clear the Character Flag (PCLF)

Octal Code: 6662
Operation: Clears the Line Printer Done Flag.

Skip on Error (PSKE)

Octal Code: 6663
Operation: Senses the content of the Line Printer Error Flag; if it contains a binary 1, indicating that an error (drum gate open, out of paper, excessive temperature) has been detected, the contents of the PC are incremented by one so that the next sequential instruction is skipped.

Load Printer Buffer, Print on Full Buffer or Control Character (PSTB)

Octal Code: 6664
Operation: Loads the character into the print buffer, and prints if the buffer is full, or if the character was a control instruction. This instruction does not clear the AC.

Set Program Interrupt Enable Flag (PSIE)

Octal Code: 6665
Operation: Sets the interrupt enable (IE) flip-flop to a one, permiting the Printer Done Flag to request a program interrupt.

Clear Line Printer Flag, Load Character, and Print (PCLF, PSTB)

Octal Code: 6666
Operation: This is a microprogram combination of PCLF and PSTB.

Clear Program Interrupt Flag (PCIE)

Octal Code 6667
Operation: Clears the interrupt enable flip-flop.

DATA COMMUNICATIONS EQUIPMENT OPTIONS

DP8-EA and DP8-EB Synchronous Modem Interface

The DP8-EB interfaces the PDP-8/E with a full-duplex or half-duplex synchronous modem for computer-to-terminal or intercomputer transfer of data. The PDP-8/E is capable of interfacing up to four communication links (channels), each having its own unique program instructions, modem interface, baud rate, priority assignment and access addresses. Data is exchanged between the PDP-8/E and the DP8-EA/EB in parallel form, using the data break facility of the computer. Data is exchanged between the DP8-EA/EB and a modem in serial form. Thus, the DP8-EA/EB performs parallel-to-serial conversion for transmit functions and serial-to-parallel conversion for receive functions. The interface also provides level conversion, character detection, modem control, and program-controlled interface with the computer.

Data exchange between the PDP-8/E and DP8-EA/EB interface is accomplished via the data break facility to or from any location within 32K of memory. Word count (WC), Current Address (CA) and character detection are performed using additional data break cycles to a specified set of locations in field zero. The DP8-E interface for one communication link consists primarily of MSI logic packaged on two PDP-8/E modules, which plug into the OMNIBUS. A cable provided with the modules mates with a connector on the modem. Two types of interface units (designated DP8-EA and DP8-EB) are available. A DP8-EA interface operates with a Bell System 200-Series Modem or equivalent, and DP8-EB operates with a Bell System 300-Series Modem or equivalent.

Specifications

Transfer Mode	Modem—Full—or half-duplex (serial data) Computer—Multi-cycle data break (parallel data)
Modem Interface	Jumper-selectable for: 1) Bipolar EIA/CCITT (RS232-C) 2) Current Mode; where MARK = 5 ma or less and SPACE = 23 ma or greater 3) TTL compatible
Baud Rate	71,000 (max)
Character Length	Jumper selectable for 6, 7, or 8 bits
Response Time	Break cycles: 1/Baud rate Program Interrupts: 1/Baud rate * bits/character (one character time)
Character Recognition	Detects four program selected characters. Flag bit stored with character determines whether program is flagged or character is stripped.

Synchronization Character	Transmit: Non-hardware function; part of data for transmission. Receive: Receive sync code is jumper-selectable. Two or more consecutive sync characters must be detected before hardware is activated.
Clock	Modem timing or tabs for customer-supplied clock

Modem Compatibility (Typical)

Type	Speed (Baud)
Bell 201A	2000
" 203A	2400
" 205B	600,1200 or 1800
" 301B	40,800
" 301B,C	19,000 to 50,000
Rixon FM-12	1200
" Sebit 48	4800
G.E. TDM Series	2400
Lenkurt 26C	120-2400

Additional Features

Break Priority	Jumper selectable for priorities 1 through 7.
Device Codes	Jumper selectable for using up to four DP8-E modules.
Access Address	Jumper selectable for up to six groups corresponding to DP8-E assignment (up to four active and two spares). Each group or interface module can have up to 16 access or file addresses.

Programming

The IOT instructions which follow control the DP8-EA/EB. For multiple-channel interfacing, the octal codes listed are used for Channel 1; IOTs containing device codes 42 and 43 are as used for Channel 2, etc., as follows:

Channel	Access Addresses (9 per channel)	IOT Device Codes
1	7720-7730	640x/641x
2	7700-7710	642x/643x
3	7660-7670	644x/645x
4	7640-7650	646x/647x
	*7620-7630	
	*7600-7610	

* These spare access addresses may be used in case of conflict with existing programs.

Access address assignments are determined by low order bits (8-11) as follows:

0000 0001 0010 0011	Test Characters
0000	Receive Word Count (WC)
0101	Receive Current Address (CA)
0110	Not used
0111	Transmit Word Count (WC)
1000	Transmit Current Address (CA)

Transmit Go (SGTT)

Octal Code: 6405
Operation: Sets the Transmit Go flip-flop.

Receive Go (SGRR)

Octal Code: 6404
Operation: Sets the Receive Go flip-flop.

Skip if Character Detected (SSCD)

Octal Code: 6400
Operation: Skips the next instruction if the character detect flag is set.

Clear Sync Detect (SCSD)

Octal Code: 6406
Operation: Clears the sync character detection flip-flop.

Skip if Receive Word Count Overflow (SSRO)

Octal Code: 6402
Operation: Skips the next instruction if the receive word count overflow flag is set, and clears the flag.

Skip if Transmit Word Count Overflow (SSTO)

Octal Code: 6403
Operation: Skips the next instruction if the transmit overflow flag is set. The flag is cleared if the skip occurs.

Clear Synchronous Interface (SCSI)

Octal Code: 6401
Operation: Initializes all active functions in the synchronous interface.

Read Transfer Address Register (SRTA)

Octal Code: 6407
Operation: Jam transfers the content of the transfer address register to AC00-11.

Load Control (SLCC)

Octal Code: 6412
Operation: Transfers the contents of AC00-02 to the DP8-E for selecting:

AC00 = Terminal ready
AC01 = Idle mode
AC02 = Synchronous interface enable

Skip if Ring Flag (SSRG)

Octal Code: 6410
Operation: Skips the next instruction if the ring flag is set, and clears the flag.

Skip if Carrier/AGC Flag (SSCA)

Octal Code: 6411
Operation: If the Carrier/AGC flag is set, skips the next instruction and clears the flag.

Read Status 2 (SRS2)

Octal Code: 6414
Operation: Jam transfers status to AC00-06.

Read Status 1 (SRS1)

Octal Code: 6415
Operation: Jam transfer status to AC00-06.

Load Field (SLFL)

Octal Code: 6413
Operation: Transfers the contents of AC00-05 to the field select registers. AC00-02 selects the transmit field; AC03-05 selects the receive field.

Skip on Bus Error (SSBE)

Octal Code: 6416
Operation: If the bus error flag is set, skips the next instruction and clears the flag.

Read Character Detected (SRCD)

Octal Code: 6417
Operation: Jam transfers the content of a two-bit register containing the address of the detected character to AC10 and AC11. The two bits of the register correspond to the two low order bits of the access address, where the characters for detection are stored.

Maintenance Instruction (also SRCD)

Octal Code: 6417
Operation: When AC00 is set, causes a single clock pulse on the external clock line to the modem. In the test configuration, this line is returned as the transmit and receive clock line, permitting single-step testing of the transmit and receive circuits.

DP8-EP Redundancy Check Option

The DP8-EP redundancy check option is designed to complement the DP8-EA and DP8-EB synchronous interface by providing parity generation and checking facilities. Vertical redundancy checks (VRC), logitudinal redundancy checks (LRC), and cycilic redundancy checks (CRC) can be performed by this option. The cycilic redundancy check is IBM-compatible CRC-12 and CRC-16.

The DP8-EP operates directly with the PDP-8/E from program-controlled instructions. Thus, when not used with the communications equipment, it can be used with other devices. The DP8-EP consists primarily of MSI logic packaged on a single PDP-8/E module, which plugs into the OMNIBUS. All control functions and character options are programmable. The primary purpose of the DP8-EP parity option is to reduce processor overhead for data communications applications where character parity (VRC) and/or Block Check Character (BCC) Accumulation (LRC or CRC) are required for error detection. The types of parity generation or checks that the DP8-EP can perform are defined below:

a. Vertical Redundancy—Parity is on a character basis where one bit slot of each character is reserved for the parity bit. Odd parity (odd number of binary ones) is generated by this option; however, capabilities are provided for checking odd or even parity.

b. Longitudinal Redundancy—This type is a BCC accumulation over a block of message characters; that is, the LRC is an accumulated EXCLUSIVE OR of all character bits (including parity bits) in a message. This method is more reliable than the VRC in detecting errors. A system may use both the LRC and VRC to increase the probability of detecting errors. Both the transmitting and receiving station must compute the BCC; at the end of the message block, the BCC Accumulations are compared at the receiving stations. If they are equal, the message is assumed to be without error.

c. Cyclic Redundancy—As implemented in this option, is IBM-compatible for CRC BCC accumulation (CRC16/12).

   The CRC check sum is the remainder derived from dividing the numerical value of the message by a constant. The division is performed serially, the quotient is discarded, and the remainder is stored. Both the transmitting and receiving stations must compute the BCC accumulation. At the end of each message block, the BCC accumulation is sent to the receiving station for comparison with the receive station check sum. If the two are equal, the message is assumed to be without error. CRC and VRC operations can be combined to increase the probability of error detection.

Specifications

Vertical Redundancy Check (VRC)	Tests or computes odd parity for up to eight-bit characters. Parity bit is either right-justified (AC11) or left-justified (AC04).

Longitudinal Redundancy	Computes or compares BCC accumulation for 6, 7, 8, 12 or 16-bit characters. Two bytes required for LRC 16.
Cyclic Redundancy Check (CRC)	IBM compatible for CRC-12 and CRC-16. Division constants used are: $X\uparrow 12 + X\uparrow 11 + X\uparrow 3 + X\uparrow 2 + X + 1$ for CRC-12, and $X\uparrow 16 + X\uparrow 15 + X\uparrow 2 + 1$ for CRC-16 where X is modulo 2.
Cycle Times	VRC (compute): 1.4 us (test) : 1.2 us CRC or LRC : 1.2 us CRC and VRC (compute): 1.4 us CRC and VRC (test) : 1.2 us LRC and VRC (compute): 1.4 us LRC and VRC (test) : 1.2 us

Programming
The instructions associated with the DP8-EP option are as follows:

Compute VRC (RCCV)

Octal Code: 6113
Operation: Transfers character in AC4-11 to DP8-EP parity register and generates odd vertical parity. Result is then jam-transferred to AC with parity bit in AC04 or AC11 as defined by the program. This instruction can be issued with the RCCB instruction.

Test VRC and Skip (RCTV)

Octal Code: 6110
Operation: Checks parity of character in AC4-11. For odd parity, the next instruction is skipped if parity of character is odd. This instruction can be issued with the RCCB instruction.

Generate BCC (RCGB)

Octal Code: 6114
Operation: Generates an LRC or CRC Block Check Character (BCC). The LRC can be generated from 6, 7, 8, 12, or 16 (two six-bit bytes) bit characters, while CRC 12/16 can be computed from 6 to 8-bit characters, respectively. BCC verification: The transmitted BCC is compared to LRC Receive BCC by treating the BCC as part of the overall accumulation. In doing so the receive BCC generator will go to zero if there were no errors in transmission.

Read BCC High (RCRH)

Octal Code: 6112
Operation: Jam-transfers the 6, 7, 8, or 12 MSBs of BCC accumulation to the AC (right-justified). The quantity of bit transferred to the AC is dependent on the BCC length selected with the RCLC instruction.

Read BCC Low (RCRL)

Octal Code: 6111
Operation: Jam-transfers the 8 LSBs of BCC accumulation to AC (right-justified). The instruction is used for the 16-bit BCC.

Clear BCC Accumulation (RCCB)

Octal Code: 6116
Operation: Clears the 16-bit BCC register.

Load Control (RCLC)

Octal Code: 6115
Operation: Jam-transfers content of AC to redundancy control register to define the operation as follows:

Bits

AC05 = 0: CRC BCC
1: LRC BCC

AC 6 7 8

0 0 0 = 16-bit BCC
0 0 1 = 12-bit BCC
0 1 0 = 8-bit BCC
0 1 1 = 7-bit BCC
1 0 0 = 6-bit BCC

AC 9 = 0: Generated parity to AC4
= 1: Generated parity to AC11

AC10 = 1: An RCGB instruction also causes a RCCV instruction sequence to accrue. The BCC accumulation will be computed with the corrected character parity.

AC11 = 1: An RCGB instruction also causes a RCTV instruction sequence to accrue.

Interface to Bell 201 Modems

The DP8-EA Synchronous Data Interface module is connected to a Bell Model 201 modem (or equivalent) by a 25-ft cable terminated at the modem end with a 25-pin male connector. Standard interface signals are bipolar (EIA/CCITT); however, current mode or TTL compatible signals can be selected using jumper options on the DP8-EA. Interface signals versus connector pin assignments are provided in Table 7-1. In addition, signal or protective ground is provided on pin 1. Signal ground is provided on pin 7.

Table 7-1
Connector Pin Assignments for Bell Series 201

Pin	Signal
2	Send Data
3	Receive Data
4	Send Request
5	Clear to send
6	Interlock
8	Carrier on-off
15	Serial Clock transmit
17	Serial clock Receive
20	Remote control
22	Ring indicator 1
24	External timing

Interface to Bell Models 301 and 303 Modems

The DP8-EB Synchronous Data Interface Module is connected to Bell 301 or 303 modems (or equivalent) by a 25-ft cable terminated at the modem end by a male 12-pin coaxial connector. Standard interface signals are the current-mode type where a MARK is 5 milliamperes or less and a SPACE is 23 milliamperes or greater. However, there are two exceptions for the 303 modem; Data Terminal Ready and Ring Indicator. These signals are normally bipolar levels (EIA/CCITT). Interface signals and connector pin assignments for the Bell 301 and 303 modems are defined in Table 7-2. In addition, protective ground is provided on pin 1 and signal ground is provided on pin 7.

Table 7-2
Connector Pin Assignments for Bell Model 301 and 303 Interface

Pin	Model 301	Model 303
E	Send Data	Send Data
K	Received Data	Receive Data
D	Send Request	Send Request
C	Clear to send	Clear to send
F (Center Conductor)	Interlock	Data set Ready
M (Center Conductor)	Carrier on-off	AGC Lock
J	Serial clock transmit	Serial Clock Transmit
L	Serial clock receive	Serial clock receive
M (Outer Conductor)	Not used	Data terminal ready*
F (Outer Conductor)	Not used	Ring indicator*
H	Serial clock transmit (external)	Serial clock transmit (external)

*Bipolar

CARD READER OPTIONS

Type CR8-E Card Reader and Control

The CR8-E Card Reader option equips the PDP-8/E computer to accept input from EIA standard data cards. It reads 12-row, 80-column punched cards at a nominal rate of 200 cards per minute photoelectrically. The control circuit for this device is located on a single PDP-8/E module, which plugs into the OMNIBUS. The card reader has an internal power supply and can be tested off-line. For table space requirements, please refer to the specification section which follows.

A select instruction starts a card moving through the read station, where all 80 columns are read on a column-by-column basis, beginning with column one. Card data may be read in any one of three modes. In the binary reading mode, the data is transferred directly from the rows of the card to bits in the AC. The top row of the card (row 12) goes into AC0 and the bottom row (row 9) goes into AC11. In the alphanumeric reading mode, the data is automatically decoded into a six-bit BCD representation and transferred into the least significant six bits of the accumulator. Use of the six-bit decoding minimizes the size of translation tables and is fully compatible with the Hollerith code as used at this time. A proposed expansion of the Hollerith code would require use of the compressed reading mode. In this mode, rows 9, 12, 11, 0, and 8 are transferred directly to AC4, AC5, AC6, AC7, and AC8, respectively, while rows 1 through 7 are decoded into three-bit BCD representation in AC9, AC10, and AC11. This decoding is based on the lack of double punches in rows 1 through 7, both in the present Hollerith and the proposed extension of Hollerith. If such a double punch is read in the compressed reading mode, the CR8-E validity checking circuitry will assert a one in AC0 (the sign bit). Regardless of the reading mode being used, a punched hole is interpreted as a binary one and the absence of a hole is binary zero.

Four program flags indicate card reader conditions to the computer. (The status of these flags may be examined by means of the RCNI instruction.) The Data Ready Flag sets, requesting a program interrupt, when a column of information is ready to be transferred into the AC. A read instruction (alphanumeric, binary, or compressed) must be issued within 1.0 ms after the Data Ready Flag sets in order to avoid data loss. The Card Done Flag sets, requesting a program interrupt, when the card leaves the read station. A new select instruction must be issued immediately after the Card Done Flag sets to keep the reader operating at rated speed.

The Ready True Transition Flag sets, requesting a program interrupt, whenever the ready lead from the card reader to the control goes true, indicating that the card reader is ready. This feature permits the computer program to perform other tasks while awaiting manual intervention to clear a card reader problem such as lack of cards. The interrupt will notify the computer when the card reader is ready to resume reading cards. The fourth flag, the Trouble Transition Flag, sets, requesting a program interrupt, whenever the ready lead from the card reader to the control goes false, indicating an error condition in the card reader. (Error condition when used here refers to a transport error, not a data error

such as an improper validity check.) This flag is cleared by initialize or by means of the Clear Transition Flags instruction.

Specifications

Size:	18 in. high; 14 in. wide; and 18 in. deep.
Weight:	52 lb.
Card Rate:	200 per minute
Input Power:	115 Vac + or − 10 Vac, 60 Hz + or − 5 Hz, single phase, 300W maximum (50 Hz unit available)
Card Specification:	The card reader is designed to read 7⅜ in. x 3¼ in. cards conforming to the material and size requirements of EIA Standard RS-292 Media 1.
Card Capacity:	Both input hopper and output stacker hold 400 cards. Cards may be added or removed during reader operation.

Programming
The following instructions are used with the CR8-E option:

Skip on Data Ready (RCSF)

Octal Code: 6631
Operation: Senses the status of the data ready flag; if it is set (indicating that information for one card column is ready to be read), the contents of the PC are incremented by one, so that the next sequential instruction is skipped.

Read Alphanumeric (RCRA)

Octal Code: 6632
Operation: Transfers the six-bit Hollerith code for the 12 bits of a card column into AC6-11, and clears the Data Ready Flag. This instruction does not detect illegal characters.

Read Binary (RCRB)

Octal Code: 6634
Operation: Transfers the 12-bit binary code for a card column directly into the AC, and clears the Data Ready Flag. Information from the card column is transferred into the AC so that card rows 12, 11, and 0 enter AC0-2 and card rows 1 through 9 enter AC3-11, respectively.

Read Conditions Out to Card Reader (RCNO)

Octal Code: 6635
Operation: Reads AC10 into a Ready True Transition/Trouble Transition interrupt enable flip-flop. If AC10 is a 1, this flip-flop is set, enabling the generation of an interrupt whenever the reader goes from not ready to ready or from ready to not ready. This flip-flop is cleared when the PDP-8/E is initialized. The RCNO instruction also reads AC11 into a flip-flop which, when set by AC11 being a 1, enables the generation of an interrupt whenever the card done or data

ready flags are raised. For program compatibility with other family-of-eight computers, initializing the PDP-8/E sets the card done/data ready interrupt enable.

Read Compressed (RCRC)

Octal Code: 6636
Operation: Transfers an eight-bit compressed code for the 12 bits of a card column into AC4-11, and clears the data ready flag. Data from row 9 goes to AC4, zones 12, 11, and 10 to AC5, 6, and 7 respectively, and data from row 8 goes to AC8. Data from rows 1 through 7 is compressed into a BCD representation in AC9, 10, and 11. Should there be more than one bit of data in rows 1 through 7 (an invalid condition), hardware validity check circuitry will read a 1 into AC0 (sign bit).

Read Conditions in from Card Reader (RCNI)

Octal Code: 6637
Operation: Status of Ready True Transition Flag, Trouble Transition Flag, Card Done Flag, and Data Ready Flag is read into AC3, AC2, AC1, and AC0 respectively.

Skip on Card Done Flag (RCSD)

Octal Code: 6671
Operation: Senses the status of the card done flag; if it is set (indicating that the card has passed the read station), the contents of the PC are incremented, skipping the next instruction.

Select Card Reader and Skip if Ready (RCSE)

Octal Code: 6672
Operation: Senses the status of the card reader; if it is ready, the contents of the PC are incremented, skipping the next sequential instruction, a card is started toward the read station from the input hopper, and the Card Done Flag is cleared.

Clear Card Done Flag (RCRD)

Octal Code: 6674
Operation: Clears the Card Done Flag. This instruction allows a program to stop reading at any point in the card deck.

Skip if Interrupt Being Generated (RCSI)

Octal Code: 6675
Operation: Senses the status of all flags. If a flag is raised and the generation of interrupts by that flag is enabled, the next sequential instruction is skipped.

Clear Transition Flags (RCTF)

Octal Code: 6677
Operation: Clears the Trouble Transition Flag and the Ready True Transition Flag.

Example Subroutine
A logical instruction sequence to read cards is the following:

```
START,  RCSE            /START CARD MOTION AND SKIP IF
                        /READY
        JMP NOTRDY      /JUMP TO SUBROUTINE THAT TYPES
                        /OUT "CARD READER MANUAL INTER-
                        /VENTION REQUIRED" OR HALTS
NEXT,   RCSF            /DATA READY?
        JMP DONE        /NO, CHECK FOR END OF CARD
        RCRA or RCRB    /YES, READ ONE CHARACTER OR ONE
                        /COLUMN AND CLEAR DATA READY FLAG
        DCA I STR       /STORE DATA
DONE,   RCSD            /END OF CARD?
        JMP NEXT        /NO, READ NEXT COLUMN
        JMP OUT         /YES, JUMP TO SUBROUTINE THAT
                        /CHECKS CARD COUNT OR REPEATS AT
                        /START FOR NEXT CARD
```

The CR8-E performs validity checking only when using the RCRC instruction. A programmed validity check can also be performed by reading each card column in both the alphanumeric and binary modes (within the 1.0 ms time limitation), and then making a comparison check.

Controls and Indicators
Before commencing a card reading program, load the input hopper with cards and press MOTOR START and READ START pushbuttons. The functions of the manual controls and indicators, as they appear from left to right, are as follows:

CONTROL OR INDICATOR	FUNCTION
ON/OFF	Toggle power switch. Applies power to all circuits except the drive motor.
MOTOR START	Momentary action pushbutton with separate indicator. Applies power to the main drive motor. This is also used as a reset to clear error indicators, and, therefore, will not operate if there is an unremedied condition such as: a. Empty input hopper b. Full output hopper c. All photo cells not lit d. Internal power supply not operational.
READ START	Momentary action pushbutton with separate indicator. Causes ready line to go

CONTROL OR INDICATOR	FUNCTION
	high, enabling card reading under control of the external read instructions. If the read instruction is being given, or the control cable is disconnected (off-line testing), card reading begins immediately at full reading speed.
READ STOP	Momentary action pushbutton with separate indicator. Inhibits further card reading until READ START is pressed again. Ready line goes low, and read stop condition is indicated. This signal does not stop the drive motor. However, a read stop condition is indicated anytime the drive motor is stopped.

CONTROL OR INDICATOR	FUNCTION
INDICATORS	
1. PICK FAIL	Lights when a card fails to enter the read station after two successive pick attempts.
2. DARK CHECK	After the card enters the read station, a check is made at hypothetical positions 0 and 81 to be sure all photocells are dark. If not, this indicator lights, and data outputs are immediately inhibited.
3. STACKER FAIL	When three cards have passed the read station and none have been stacked, this indicator lights, preventing more than three cards from being in the track at once.
4. HOPPER EMPTY	Indicates that the input hopper is empty.
5. STACKER FULL	When approximately 400 cards are in the stacker hopper, this indicator lights.
6. SYNC FAIL	This indicator lights if the sync signal is lost. Internal timing signals are derived from an oscillator that is synced to the track speed.
7. LIGHT CHECK	The photocells must always be illuminated except during the time a card is being read. The Light Check Detector is inhibited each time a card enters the read station until position (count of) 84 is reached. If it fails to leave the read station by that time, this indicator lights.

NOTE

The CR8-E and the CM8-E cannot be used on the same system, because they share IOT codes.

Type CM8-E Optical Mark Card Reader and Control

The CM8-E Optical Mark Card Reader option permits the PDP-8/E computer to accept information from marked or punched data cards with timing marks at a nominal rate of 200 cards per minute. It reads 12-row, 40-column mark sense cards and 12-row, 40-column punched cards. The control circuit is located on a single PDP-8/E module that plugs into the OMNIBUS.

A select instruction starts a card moving through the read station, where all 40 columns are read on a column-by-column basis, beginning with column one. Card data may be read in any one of three modes. In the binary mode, the data is transferred directly from the rows of the card to bits in the AC. The top row of the card (row 12) goes into AC0 and the bottom row (row 9) goes into AC11. In the alphanumeric mode, the data is automatically decoded into a six-bit BCD representation and transferred into the least significant six bits of the accumulator. Use of the six-bit decoding minimizes the size of translation tables and is fully compatible with the Hollerith code as used at this time. A proposed expansion of the Hollerith code would require use of the compressed reading mode. In this mode, rows 9, 12, 11, 0, and 8 are transferred directly to AC4, AC5, AC6, AC7, and AC8, respectively, while rows 1 through 7 are decoded into three-bit BCD representation in AC9, AC10, and AC11. This decoding is based on the lack of double punches in rows 1 through 7, both in the present Hollerith and the proposed extension of Hollerith. If such a double punch is read in the compressed reading mode, the CR8-E validity checking circuitry will assert a one in AC0 (the sign bit). Regardless of the reading mode being used, a punched hole or a non-reflective mark is interpreted as a binary one and the absence of such a hole or mark is interpreted as a binary zero.

The flag and interrupt facilities are identical to those of the CR8-E.

Specifications

Card Rate:	200 per minute
Input Power:	115 VAC + or − 10 VAC, 60 Hz + or − 5 Hz, a single phase, 300W maximum (50 Hz unit available)
Card Specification:	The card reader is designed to read 7⅜ in. x 3¼ in. optical mark cards conforming to the material and size requirements of EIA Standard RS-292 Media 1. Format and printing requirements are specified in the DEC Mark Sense Card Specification.
Card Capacity:	Both input hopper and output stacker hold 400 cards. Cards may be added or removed during reader operation.

Programming

The instruction set and example subroutine set forth for the CR8-E also applies to the CM8-E.

LABORATORY PERIPHERALS

AD8-EA Analog-to-Digital Converter

The AD8-EA converter is a 10-bit successive-approximation type with sample and hold circuits, conversion circuits, an input buffer, and con-

trol logic contained on one PDP-8/E module. The converter can be used singularly with one channel input having an input range from 0 to +10 volts or can be used with AM8-EA and AM8-EB Multiplexers to perform conversion for up to 64 channels having full-scale inputs of + or − 1 volt. Analog inputs are connected to the module by H851 edge connectors from the multiplexer or by a shielded twisted pair from an external device.

Operation of the AD8-EA converter is controlled by IOT instructions. A conversion is initiated by an ADCV instruction or an input from the AM8-EA multiplexer, or from the Real Time Clock DK8-EP. An input starts the conversion and clears the A/D Done Flag. When the conversion is complete, the converter sets its A/D Done Flag. This flag is sensed by an ADSF instruction. If set, it causes the next instruction to be skipped so that the 10-bit digital word can be transferred to AC2-11 by an ADRB instruction. Since the 10-bit word is in two's complement form, AC00 and AC01 copy AC02 (sign-extended format). The converter contains an interrupt enable flip-flop that is controlled by program instructions. When set by an ADEI instruction or by INITIALIZE, this flip-flop permits the converter to generate interrupt requests to the program interrupt facility upon completion of a conversion. When cleared by an ADDI instruction, this flip-flop prevents generation of interrupt requests.

The converter also contains circuits for detection and sensing of a timing error. A timing error is defined as the receipt of a conversion request while a conversion is in progress. If this condition occurs, a Timing Error Flag is set. The Timing Error Flag is sensed by an ADSK instruction. An ADCL instruction clears this flag. This instruction also clears the A/D Done Flag so that another conversion can be implemented.

Specifications

Input Voltage Range:	0 to +10v
Input Impedance:	Greater than 70K ohms in parallel with 300 pf
Output Format:	Parallel: 10 bits right-justified and sign extended, two's complement
Resolution:	+ or − ½ LSB at 20 degrees C
Conversion Rate:	approximately 50 kHz
Sample Acquisition Time:	5 μs + or − 1 μs
Aperture Time:	200 ns
Common Mode Rejection:	35 db (Typical)
Common Mode Voltage:	+ or − 3.5v (from system fault line ground)
Overvoltage Recovery Time:	8 μs

Programming

The following instructions are associated with converter operation:

Set A/D Converter Interrupt Enable (ADEI)

Octal Code: 6530

Operation: Sets the converter interrupt enable flip-flop to permit generation of an interrupt request when conversion is complete.

Skip on A/D Done Flag (ADSF)

Octal Code: 6531
Operation: Senses the A/D converter Done Flag; if it is a one, the next instruction is skipped so that the converter can be serviced.

Clear A/D Done and Timing Flip-Flop (ADCL)

Octal Code: 6532
Operation: Clears the A/D Done flag and timing flip-flops to ready the converter for another conversion.

Skip on A/D Timing Error (ADSK)

Octal Code: 6533
Operation: Senses the timing error flag; if it is a one (indicating that a second conversion request occurred while conversion was in progress), the next instruction is skipped so that a JMS to an error routine can be implemented.

Read A/D Converter Buffer (ADRB)

Octal Code: 6534
Operation: Jam transfers the content of the converter buffer to AC2-11 and clears the A/D Done Flag. AC00 and AC01 copy AC02.

Convert Analog Voltage to Digital Value (ADCV)

Octal Code: 6535
Operation: Clears the A/D Done Flag and starts the analog-to-digital conversion. When the conversion is complete, sets the A/D Done Flag.

Read A/D Buffer and Start Conversion (ADST)

Octal Code: 6536
Operation: Jam transfers the contents of the A/D buffer to AC2-11. AC00 and AC01 copy AC02. Clears the A/D Done Flag and starts another conversion.

Clear Interrupt Enable F/F (ADDI)

Octal Code: 6537
Operation: Clears the A/D Interrupt Enable Flip-Flop to prevent interrupt requests from being generated by the A/D converter.

A program service routine for the A/D converter can be written as follows:

```
ADCV        /CLEAR A/D DONE AND START CONVERSION
ADSK        /SKIP IF A/D TIMING ERROR FLAG = 1
SKP         /SKIP IF TIMING ERROR FLAG NOT = 1
JMS ERROR   /ERROR ROUTINE
ADSF        /SKIP IF DONE FLAG = 1
```

```
JMP .—1          /WAIT
ADRB             /READ A/D BUFF.
  .
  .
  .
  .
```

AM8-E 8-Channel Analog Multiplexer

The AM8-E is an 8-channel analog multiplexer designed for use with the AD8-EA A/D Converter. The multiplexer accepts unipolar positive or negative analog input having a full-scale range of 1v and converts these inputs into a full-scale +10v unipolar output supplied to the AD8-EA Converter.

The multiplexer control, type AM8-EA consists of:

a. Capabilities for addressing up to 64 channels
b. IOT control
c. Load/read logic contained on two PDP-8/E modules, which plug into the OMNIBUS.

The AM8-EB consists of multiplexer switches and scaling amplifiers for 8 analog channels. The AM8-E can be expanded up to 64 channels in eight-channel groups by adding additional AM8-EB 8-channel Multiplexer Expander Modules. Multiplexer operation is controlled by a versatile set of IOT instructions. These instructions and the associated multiplexer control provide the capabilities for random or sequential selection of channels, combining the operation of the A/D converter with that of the multiplexer and reading the active channel address to the computer. Two programmable address modes are provided: autoincrement or nonautoincrement. The AM8-E is set to nonautoincrement mode when INITIALIZE is generated.

For nonautoincrement operation, the computer resets the AUTO MODE flip-flop with an AMRM instruction. The computer then issues an AMLC or AMMS instruction. The AMLC instruction simply loads the channel address into the multiplexer address register. This instruction must be followed by another instruction either to the multiplexer or the A/D converter. In contrast, the AMMS instruction selects the channel, transfers the result of the previous conversion from the A/D converter buffer to the computer and starts a conversion for the selected channel. Two instructions (AMIM and AMIC) can be used with the nonautoincrement mode to increment addresses for sequential channel sampling under program control. The AMSM instruction only increments the address: the AMIC instruction increments the address and transfers the previous conversion to the computer.

In the autoincrement mode, channel addresses are incremented automatically at the completion of a conversion by an A/D Done output from the converter. An AMSM instruction selects this mode by setting the AUTO MODE flip-flop. The computer specifies the first address of interest by issuing an AMLC instruction, then issues A/D converter instructions to start an A/D conversion and read the result of the previous conversion. Upon completion of a conversion, the A/D Done input in-

crements the multiplexer channel address for another sample. This process continues until the AUTO MODE flip-flop is reset by an AMRM instruction.

Specifications

Input Voltage	Unipolar, 0 to — 1V or 0 to + 1V
Input Impedance	70K ohms + or — 2%, shunted by 300 pf
Output	Unipolar, + 10V full scale
Common Mode Rejection	Greater than 25 dB, 35dB typical
Overload Protection	+ or — 67V from fault line (indefinitely)
Overload Recovery Time	8 ms
Frequency Response	Flat from 0 to 30 KHz, —3dB 60 KHz
Leakage Current	Negligible at 70K ohms impedance
Long Term Stability (1 hour)	1%

Programming
The following instructions are associated with AM8-EA operation:

Load Multiplexer Channel (AMLC)

Octal Code: 6541
Operation: Loads channel address into multiplexer register from AC06-11.

Increment Multiplexer Channel (AMIM)

Octal Code: 6542
Operation: Increments multiplexer channel address for sequential sampling of channels. Sets channel address to 0 after last implemented channel is sampled.

Read Multiplexer Channel (AMRC)

Octal Code: 6543
Operation: Transfers the multiplexer channel address to AC06-11 using a jam transfer.

Set Multiplexer To Auto-Increment Mode (AMSM)

Octal Code: 6544
Operation: Selects auto-increment mode for sequential sampling of channels. Address is incremented by A/D Done from converter.

Reset Auto-Increment Mode (AMRM)

Octal Code: 6545
Operation: Clears the auto-increment mode control so that A/D Done will not increment channel address.

Load Multiplexer Channel (AMMS)

Octal Code: 6546
Operation: Loads content of AC6-11 into multiplexer channel address register. The contents of A/D buffer are jam transferred to AC2 through AC11. AC0 and AC1 copy AC2. The Done Flag is cleared and another conversion is started.

Increment Multiplexer Channel (AMIC)

Octal Code: 6547
Operation: Increments multiplexer register channel address and reads previous A/D conversion value into AC0-11.

A program for autoincrement mode of operation can be written as follows:

```
        AMSM          /SET MODE TO INCREMENTING
        CLA
        TAD ADRS
        AMLC          /LOAD ADDRESS OF FIRST CHANNEL
        STA
        TAD STORE
        DCA 10        /SET UP STORAGE AREA
        TAD NBR
        DCA CTR       /AND CONVERSION COUNTER
CONV.   ADCV          /CONVERT
        ADSF
        JMP .—1
        DCA I 10      STORE CONVERTED VOLTAGE
        ISZ CTR
        JMP CONV.     /GO BACK FOR NEXT
          •           /JOB DONE
          •
          •
          •

NBR.    ABC           /TWOS COMPLEMENT OF NUMBER OF
                      /CONVERSIONS
STORE.  DEF           /LOCATION WHERE NUMBERS WILL BE
                      /STORED IN MEMORY
```

A program for nonautoincrment mode operation can be written as follows:

```
        AMRM          /RESET AUTO MODE F/F
RETURN. AMMS          /LOAD MUX CHANNEL, READ A/D BUFFER
          •           /CLR A/D DONE FLAG AND START
                      /CONVERSION
                      /STORE RESULT, FIGURE AND START
                      /CONVERSION
          •           /NEW ADDRESS IN AC
        ADSF          /SKIP IF A/D DONE = 1
        JMP  —1       /WAIT
        JMP RETURN    /
```

DR8-EA 12-Channel Buffered Digital I/O

The DR8-EA Digital I/O can be used to control 12 discrete digital switching circuits located externally, and can be used to accept 12 discrete digital inputs from external sources. The unit consists of IOT control logic, a 12-bit input buffer, a 12-bit output buffer, and 3 multiplexer ICs that control the flow of data for input and output operations. All circuits are TTL logic and are mounted on a single PDP-8/E module which plugs into the OMNIBUS. The standard TTL outputs are connected to the external load via two H854 connectors on the module. Inputs from external sources are also connected to the DR8-EA using H854 connectors.

Data outputs are updated under program control. Standard output drivers have a TTL 30-unit load capability. These outputs are protected against short circuits and negative transitions greater than 0.6V. For an output function, the computer issues a DBOR, DRZR, or DBAN instruction. For DBOR instructions, only logical ones in the AC are loaded into the output register; AC bits containing logical zeros do not affect output register bits. For DBZR instructions, logical ones in AC result in logical zeros in corresponding bits of the output register; again, logical zeros in the AC do not affect output register bits. For DBAN instructions, the content of the AC is transferred to the output register.

Data inputs must be TTL compatible, have a positive transition to +2.6V or greater for a logical one, and have a pulse duration of greater than 50 ns. Pulse rise and fall time should be less than 150 ns for maximum noise immunity. The input register bits, once set by the data inputs, remain set until read by a DBIN instruction. When this IOT instruction is issued, the content of the input register is gated to the AC via the OMNIBUS. A DBCL instruction, used with DBIN instruction, enables inputs that occurred too late to be read by the last DBIN instruction to remain in the input register to be read by the next DBIN instruction. Correct usage of this feature results in "zero dead time" for events. Input lines are clamped to protect the input stages from negative voltage in excess of 0.6V. The data inputs may be read directly (bypassing the Input Register) by jumpers on the module.

Specifications

Input Format	Parallel, 12 bits
Input Levels	Compatible TTL levels. Input lines clamped at 0.6V for negative input protection.
Input Connections and Pulse Width	Inputs connected to clock input of D-type flip-flop. Pulse with positive transition to 2.6V or greater and pulse width greater than 50 ns sets input register. Each input is two TTL unit loads. Pulse rise and fall time should be less than 150 ns. Optional inputs bypass the flip-flop for direct interrogation of input line status.

Output Format	Parallel, 12 bits.
Output Levels	TTL-compatible levels capable of driving 30 unit loads. Output lines are protected from short circuits to ground.

Programming

The following instructions are used for DR8-EA operation:

Output Register to AC (DBJO)

Octal Code: 6570
Operation: Jam transfers the content of the Output Register into the AC Register.

Load Ones In Output Register (DBOR)

Octal Code: 6571
Operation: Logical ones in AC set corresponding stages of Output Register. Logical zeros in AC do not affect Output Register stages.

Load Zeros in Output Register (DBZR)

Octal Code: 6572
Operation: Logical ones in AC reset corresponding stages of Output Register. Logical zeros in AC do not affect Output Register.

Load Output Register (DBAN)

Octal Code: 6573
Operation: Loads content of AC into Output Register (Jam transfer).

OR Output Register with AC (DBOO)

Octal Code: 6574
Operation: ORs content of the Output Register with AC0-11 and loads the result into AC.

Clear Input Register Stages (DBCL)

Octal Code: 6575
Operation: Logical ones in AC clear or reset corresponding bits of the Input Register. Logical zeroes in AC do not affect Input Register.

OR Input Register with AC (DBOI)

Octal Code: 6576
Operation: ORs content of Input Register with AC and loads the result into AC.

Input Register to AC (DBJI)

Octal Code: 6577
Operation: Jam transfers the content of the Input Register into the AC Register.

Laboratory Peripheral Panel

The laboratory peripheral panel is designed for compact yet versatile packaging of modular accessory equipment for laboratory environments. The panel is a 19-inch rack-mounted unit with H945-AA panel mounting frame and housing that accepts plug-in type modules or module panels. Modules can be single-width, double-width, or other multiples of single-width, and may contain a printed circuit card mounted on the vertical dimension. Controls and input/output connectors for peripheral equipment are mounted on the module front panel. Modules or module panels are attached to the panel frame using one fastener at the top and bottom of the module panel.

The following options are available:

H945-AA— Housing for mounting laboratory peripherals. Includes space for 12 panel units.

H945-BA— Blank filler panel, 1 panel unit.

H945-BB— Blank filler panel, 2 panel units.

DK8-EF— Optional panel for Real Time Clock Type DK8-EP. Contains six 3-conductor phone jacks for event inputs and outputs. It also contains 3 3/4-turn potentiometers for voltage adjustments and 3 rotary switches for line or plus and minus voltage references.

AM8-EC— 4-channel A/D panel used for AM8-EA/EB multiplexer inputs. Panel contains four 3-conductor phone jacks and four 10-turn vernier controls. Panel requires 2 panel units.

AM8-ED— 8-channel A/D panel used for AM8-EA/EB multiplexer analog inputs. Panel contains eight 3-conductor phone jacks and requires 2 panel units.

VR03-A— Model 602 Tektronix oscilloscope and mounting hardware which includes a H945AA. Panel requires 6 panel units.

SECTION 4 EXTERNAL BUS INPUT/OUTPUT EQUIPMENT OPTIONS

The following equipment options are provided for the External Bus. Positive I/O Bus Interface (KA8-E) is a prerequisite for I/O transfer and the Data Break Interface (KD8-E) is a prerequisite for any data break peripheral. Refer to Chapter 10 for the External Bus Interface Discussion.

EXTERNAL BUS INTERFACE CONTROL OPTIONS

The external bus interface options enable the PDP-8/E user to interface PDP-8/I and PDP-8/L type peripherals (such as mass storage devices, data acquisition, and control equipment) with the PDP-8/E. It also permits user-designed equipment to be interfaced with the 8/E external bus through the use of a general-purpose interface unit. A type KA8-E Positive I/O Bus Interface unit is required for any type of peripheral connected to the external bus. A type KD8-E Data Break Interface unit is required for each external peripheral that uses the data break facilities of the computer. These interface units and the BB08-P General Purpose Interface are described below. The detailed relationships of programmed I/O transfers and data break transfers are described in Chapter 10.

KA8-E Positive I/O Bus Interface

The KA8-E option enables the PDP-8/E user to interface PDP-8/I and PDP-8/L type peripherals with a PDP-8/E. This option converts OMNIBUS signals into positive programmed I/O bus signals used by PDP-8/I and PDP-8/L type peripherals. For example, 8/I and 8/L type peripherals require IOP pulses to perform their operations. The OMNIBUS does not generate internal IOP pulses, but does provide signals (MD bits 9-11) that can be converted to IOP pulses. Other signals normally required for programmed I/O transfers are also available on the OMNIBUS. The KA8-E merely buffers these signals and makes them available to the external bus at the correct time. Similarly, the KA8-E buffers peripheral inputs and makes them available to the OMNIBUS. A detailed description of the external bus interface, including signals, levels, timing relationships, and other interface data, is provided in Chapter 10.

Only one KA8-E can be used per machine. This module is required both for programmed I/O transfers and for external bus data break transfers. The KA8-E is also required whenever the BB08-P General Purpose Interface option is used, and when user-designed or user-installed logic is to be connected to the external bus. The KA8-E Positive I/O Bus Interface is contained on one PDP-8/E module that plugs into the OMNIBUS.

See Chapter 10 for details of interfacing with the external bus.

BB08-P General Purpose Interface Unit

The BB08-P General-Purpose Bus Interface provides the PDP-8/E user with the capability of interfacing user-designed or user-installed logic with the PDP-8/E external bus. (The KA8-E Positive I/O Bus Interface module is a prerequisite for using the BB08-P.)

The BB08-P can interface one receive (input) and one transmit (output) device, or two receive, or two transmit devices, and control related transfers from program instructions. In addition, the unit can supply operating power for the user's device.

The BB08-P logic is housed in one prewired DEC type H943 Mounting Panel with a self-contained Type H716 Power Supply. There are 34 module sockets not used by option modules; thus, these sockets are available for user logic modules. The spare sockets, located in two adjacent rows, can accommodate 34 single-height modules, or 17 double height modules or combinations.

The basic data format for transfers is 12-bit parallel. The organization of fields within this format is at the user's discretion; however, user logic must operate according to the following rules:

a. Data user logic to computer, via BB08, must take inverted positive-bus form:
   0V (L) = logic true (1);
   +3V (H) = logic false (0).
b. Data from computer to user logic, via BB08, must be accepted in true positive-bus form:
   +3V (H) = logic true (1);
   0V (L) = logic false (0).

c. User logic must provide pulses to the BB08 to set the Transmit and Receive flags as required. These pulses must take the form of OV to +3V transitions of not less than 100 ns duration. Rise and fall times of these pulses should be 150 ns or less.

The user may, at his discretion, use any or all of the following spare logic gates on modules of the BB08 option:

a. Four C/D flip-flops on the M216 or M206 Module at panel location A06.
b. Two TTL logic inverters on the M111 Module at panel location A07.
c. Eight open-collector bus drivers on the M623 Module at panel location 805.
d. Eight TLL two-input NAND gates on the M113 Module at panel location B06.

The BB08-P receive section consists of 12 level-converter gates, a device selector, and Receive Flag circuits. For transfers to the computer, the user device sets the Receive Flag to initiate a program interrupt for servicing the device. The computer then interrogates the skip chain by issuing Skip-on-External-Flag instructions. When a skip instruction with 37 (octal) is detected by the BB08, this device returns a skip pulse that causes a conditional jump in the computer. The program then clears the Receive Flag by issuing an IOT 6372 (octal) and transfers the input word to the computer with a 6374 (octal) instruction.

The BB08-P transmit section consists of 12 level-converter gates, a device selector, and Transmit Flag circuits. The most important difference between transmit and receive logic is that transfer from the buffered accumulator bus to the user's device is enabled whenever the BB08 device selector decodes 636X (octal).

Specifications

Data Format	12-bit parallel. Can be discrete bits or any organization of fields.
Receive/Input	TTL compatible of the inverted positive bus form: 0V (L) = logic 1 +3V (H) = logic 0
Transmit/Output	TTL compatible of the true positive bus form: +3V (H) = logic 1 0V (L) = logic 0
Power Supply Outputs (Available for User Logic)	3A at +5V 1.3A at + 15V
Power	115VAC, 60 Hz, 1A

Programming

The following instructions are used for BB08-P operation:

Skip On Transmit Flag (GTSF)

Octal Code: 6361
Execution Time: 2.6 μs
Operation: Skips the next instruction if the Transmit Flag is set.

Clear Transmit Flag (GCTF)

Octal Code: 6362
Execution Time: 2.6 μs
Operation: Resets the Transmit Flag.

(User Designated)

Octal Code: 6364
Execution Time: 2.6 μs
Operation: This instruction is not used by BB08-P; however, the BB08-P decodes this IOT to make IOP4 available to user. User can use the IOP4 pulse to strobe data into his device.

Skip On Receive Flag (GRSF)

Octal Code: 6371
Execution Time: 2.6 μs
Operation: Skips the next instruction if Receive Flag is set.

Clear Receive Flag (GCRF)

Octal Code: 6372
Execution Time: 2.6 μs
Operation: Resets the Receive Flag.

Read Device Buffer (GRDB)

Octal Code: 6374
Execution Time: 2.6 μs
Operation: Transfers data from receive device to AC0-11.

KD8-E Data Break Interface

The KD8-E Data Break Interface option provides the PDP-8/E user with the one- and three-cycle data break facilities of the computer. Each KD8-E implements one of the 12 available data break channels of the PDP-8/E.

Each KD8-E contains the hardware to implement one standard data break channel and logic for establishing multiplexing priority between break devices. The KD8-E option is contained on one PDP-8/E module that plugs into the OMNIBUS.

Data break operations and the relationships of the KD8-E for these operations are described in detail in Chapter 10. Transfer time is 1.4 microseconds (715 kHz) for the single-cycle data break devices and 4.2 microseconds for 3-cycle data break devices.

RANDOM ACCESS DISK DEVICES

RK8 Disk System

The RK8 Disk System provides the PDP-8/E user with a modular, random-access, mass-storage device that utilizes removable disk cartridges as a storage medium. A basic system consists of one RK01 Disk Drive and Control and one RK08-P Disk Interface Control housed in a DEC H950 cabinet with self-contained power supplies, control, and indicator panel. The basic system provides the capability of storing 831,488 12-bit words and is readily expandable to over 3.3 million words in increments of 831,488 words. The RK08-P controls up to four RK01 units; thus, the expansion requires the addition of up to three RK01s.

Complete write protection is provided either on a sector basis, under manual/program control, or on whole disk basis, under manual control. This feature allows the programmer complete flexibility. Sectors of the disk containing system programs can be write-protected, while other sectors, containing data or new programs under development, can be write-permitted.

Sector numbers can be sequentially located on the disk, or scattered throughout a track. This feature allows for maximum throughput. Staggering of sector numbers throughout the track allows for computation time between data block transfers and enables the most efficient use of the system.

Complete transfer rate optimization is available. Consecutive data blocks can be transferred every 5 ms without specifying continuation. This feature eliminates the need to wait one disk revolution between blocks. A complete 4K of data can be transferred in just 80 ms.

Complete track verification is performed with a preformatted cartridge. A separate leader containing track, surface, and sector address is written and verified by the format program, and checked on each transfer. This feature ensures positive track identification before a transfer is enabled.

Complete self-checking maintenance features are built in. These features permit fast and easy diagnosis of any errors.

Data transfers are performed using the single-cycle data break facilities of the PDP-8/E; thus, the KA8-E External Bus Interface and KD8-E Data Break Interface are prerequisites for using the RK8 on the PDP-8/E external bus.

Specifications

Disk Capacity:	
Storage Capacity:	Each RK01 Disk Cartridge stores 831,488 (includes three spare tracks) 12-bit words.
Expansion:	Four RK01 drives can be controlled by one RK08 control for a total of 3,325,952 words.
Data Tracks:	200 (plus three tracks)
Words per Track:	4096 (2048 words on 0 and 1 surfaces)
Sectors:	16

Words per Sector:	256
Minimum Block Size:	256
Maximum Block Size:	4096

Read/Write Heads

Type = Tunnel erase
Number = Two (one per disk surface)

Recording Parameters

Recording Method:	Double Frequency—Time plus data
Density:	704 bits/inch (outer track) to 1026 bits/inch (inner track)
Speed:	1500 + or − 30 rpm
Transfer Path:	Single cycle Data Break
Transfer Rate:	16.7 μs per word
Minimum Access Time:	2 ms step plus 37 ms settle time (Adjacent Tracks)
Average Access:	134 ms (includes settle time)
Maximum Access:	441 ms (includes settle time)
Latency Maximum:	40 ms (one revolution)
Latency Average:	20 ms (½ revolution)
Program Interrupt:	Transfer Done Flag Track Found Flag (Enabled Separately) Error Flag (Enabled Separately)
Write Lock:	The two words at the beginning of each track contain the status information that is automatically checked for write lock of sectors.

RK01 Disk Drive and Control

The RK01 Disk Drive and Control contains the drive electronics and mechanism for accepting and releasing the disk, positioning the read/write heads, and controlling reading and writing of data from RK08-P commands.

The recording medium is a removable disk cartridge mounted in a protective case. The disk is an aluminum platter coated on both sides with magnetic oxide. When the cartridge is inserted into the drive mechanism, a read/write access door is automatically opened to permit positioning of the read/write heads over the disk surfaces. The disk is retained on its drive spindle by a magnet, and is driven counterclockwise at a uniform rate of 1500 rpm by a hysteresis synchronous drive motor. Information is recorded on or retrieved from both surfaces of the disk by the upper and lower read/write heads. These heads are cushioned from the disk surface by a film of forced air which keeps them between 125 and 160 microinches from the disk surface.

The upper surface of a disk mounted on the drive spindle is normally designated surface 0; the lower surface 1. Each disk surface is divided into 203 tracks (200 data tracks and 3 spares), with track 000 on the outer periphery and track 202 on the innermost portion of the recording area.Thus, for 203 combinations, eight bits must be allotted to address a track. The interval required for moving the read/write heads from one track to an adjacent track is approximately 2 ms; this means that, for a

move between the two possible extremes (track 000 and track 203), approximately 443 ms (including settling time) is required.

Each track is divided into 16 sectors with sectors 0 (octal) through 7 (octal) on the upper or 0 surface and sectors 10 (octal) through 17 (octal) on the lower or 1 surface. Sector assignments can be staggered as desired for the most efficient operation. Four bits (three for sector and one for surface) must be allotted for selection of a sector. A sector provides storage capabilities for two 12-bit header words, 256 12-bit data words and a 12-bit parity word. The first header word defines the track and sector address and the second word stores sector protect and status bits. The 12-bit parity word defines the longitudinal parity of the data words.

Information is recorded on the disk using the double-frequency or self-clocking method, in which a composite signal (generated from clock and data signals) is supplied to the read/write electronics. The clock frequency (720 kHz) determines the basic recording rate and cell width. For the recording of a logical zero, only the clock frequency is present in the composite signal and the results in a single change of direction in the magnetic flux pattern. However, for the recording of a logical one, a data pulse, occurring at twice the clock rate, is combined with the clock signal. This composite signal produces two changes in direction of the flux pattern. Thus, a component of the clock frequency is always present for timing purposes.

RK08-P Disk Interface Control

The RK08-P interfaces up to four RK01 Disk Drive and Control units with the external bus of the PDP-8/E. As part of this function, the RP08-P:

a. Decodes programmed IOT instructions.
b. Accepts and stores word count, current address, and command and address words.
c. Selects the program-designated RK01 and starts the program-designated operation.
d. Jointly (with the selected RK01) locates the disk address (track sector and surface).
e. Generates interrupt and break requests to initiate a single-cycle data break.
f. Buffers the input/output data and performs related conversion (serial-to-parallel and parallel-to-serial).
g. Performs housekeeping chores for single-cycle data break transfers (incrementing of word count and current address and providing address for transfer).
h. Generates and checks longitudinal parity for the sectors.
i. Provides flags for denoting current conditions and error status.
j. Provides logic for aiding in maintenance of an RK8 system.

Data transfers between the disk and core memory are implemented using the single-cycle data break facilities of the computer. (Refer to Chapter 10 for a detailed description of single-cycle data break.) For this operation, the computer specifies the number of words to be transferred (word count), and loads this information into the RK08-P word count

register by issuing a DLWC instruction. The first core memory address involved in the transfer is then loaded into the current address register by issuing a DLCA instruction. Next the computer loads a command word defining the selected disk, extended memory address (if this option is used), the interrupts it will accept, and the type of operation to be performed. This information is loaded into the command register of the RK08 by a DLDC instruction. Next the program loads the disk address (track, sector, and surface address) into corresponding registers in the RK08. This information defines disk starting location. If a read operation is to be performed, a DLDR instruction is issued to load the disk address and start the operation. Similarly, for a write operation, a DLDW instruction is issued. When the correct data track is located, a data break request is generated and the data break interface logic assumes control of the transfers. For this phase, the data break logic controls the loading and conversion of disk data words by asserting a B BREAK line to the RK08-P. Information read from the disk is converted from serial form for parallel transfer to the core memory.

This information is routed via external bus lines DATA00-11, gated by the Data Break Interface, and supplied to the MB via OMNIBUS lines DATA0-11. For write operations, information is accessed at the memory location specified by the current address. Data is provided to the RK08-P via the MD00-11 OMNIBUS lines and the Positive I/O Bus Interface (where it is buffered), and supplied to the RK08-P via the BMB00-11 external bus lines. The data is loaded into an RK08-P buffer under data break control and is shifted out to the disk read/write electronics. For each word transfer, the core memory address is specified by the current address register, and then the current address and word count registers are incremented. When the proper number of words have been transferred, as denoted by word count overflow, the RK08 generates an interrupt request. When the computer honors this request, it senses the transfer done flag of the RK08 with a DSKD instruction. If this flag is set, the computer senses the peripheral error flag with a DSKE instruction to determine if any errors were recorded during the transfers. If not, the transfer is assumed to be valid.

Programming

The following instructions are associated with the RK08-P control and the RK01 disk drive and control:

Load Disk Address (DLDA)—(Maintenance Only)

Octal Code:	6731
Execution Time:	2.6 μs
Operation:	Loads the Disk address for maintenance or diagnostic functions.

Load Command Register (DLDC)

Octal Code:	6732
Execution Time:	2.6 μs
Operation:	Loads the contents of AC0-11 into the Command Register and clears AC. The Command Register bit usage is shown in Figure 7-3.

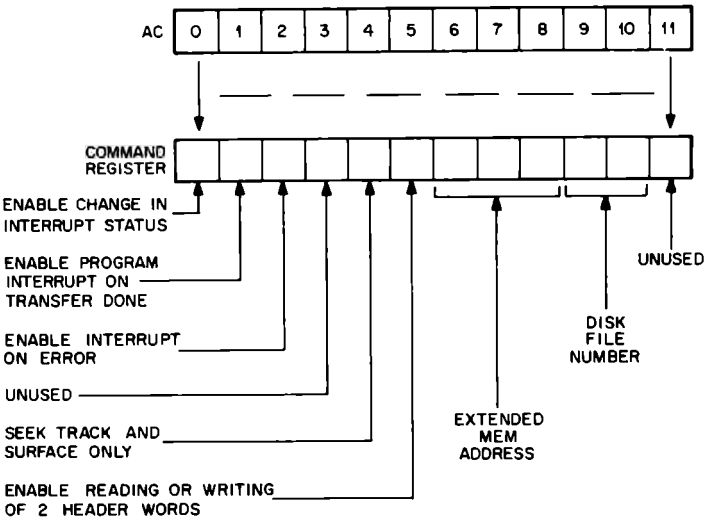


Figure 7-3 Command Word

Load Disk Address and Read (DLDR)

Octal Code: 6733
Execution Time: 3.6 μs
Operation: Loads track, surface, and sector address from AC, then clears AC. Starts read from disk if Command Register bit 4 is zero. If bit 4 is a one, instruction is executed only to seek track and surface. The relationship of bits in the AC transfer is shown in Figure 7-4.

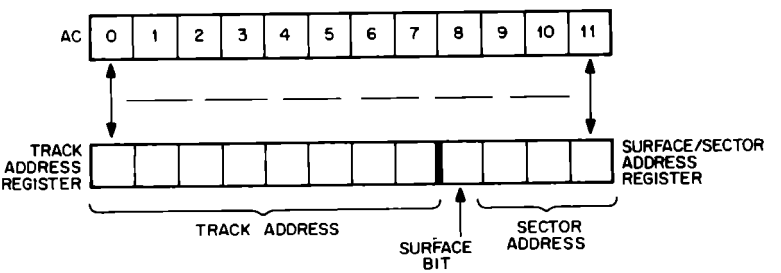


Figure 7-4 Address Word (Read)

Read Disk Address (DRDA)

Octal Code: 6734
Execution Time: 2.6 μs
Operation: Clears AC and reads content of Track Address Counter and Surface/Sector counters into AC0-11.

Load Disk Address and Write (DLDW)

Octal Code: 6735
Execution Time: 3.6 μs
Operation: Loads track, surface, and sector address from AC, then clears AC. Also starts a write operation if Command Register bit 4 is zero. If bit 4 is a one, instruction is executed only to seek track and surface.

Read Disk Command Register (DRDC)

Octal Code: 6736
Execution Time: 3.6 μs
Operation: Clears AC, then reads content of Command Register to AC0-11.

Load Disk Address and Check Parity (DCHP)

Octal Code: 6737
Execution Time: 4.6 μs
Operation: Loads track, surface, and sector address from AC, then clears AC. If bit 4 of Command Register is zero, reads data at specified address and checks parity. If bit 4 is one, instruction is executed to seek track and surface.

Read Disk Status Register (DRDS)

Octal Code: 6741
Execution Time: 2.6 μs
Operation: Clears AC, then reads content of Status Register into AC. Status bits are defined in Figure 7-5.

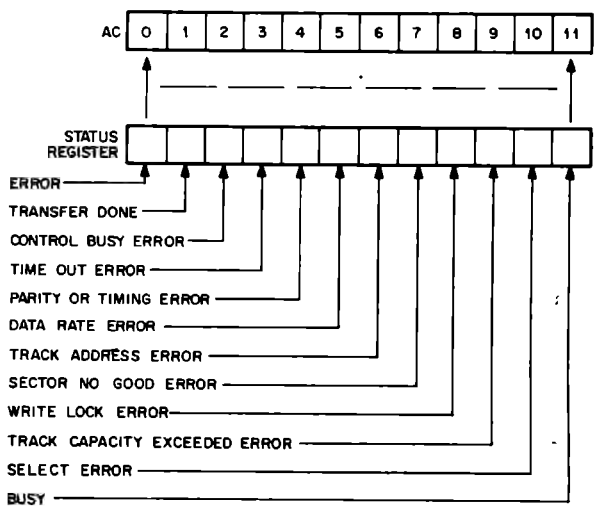


Figure 7-5 Status Word

Clear Status Register (DCLS)

Octal Code: 6742
Execution Time: 2.6 μs
Operation: Clears Status Register.

Load Maintenance Register (DMNT)

Octal Code: 6743
Execution Time: 3.6 μs
Operation: Loads the content of AC into Maintenance Register and performs specified operation. Bits remain in Maintenance Register until DMNT is issued with AC bits = 0, Maintenance functions are defined in Figure 7-6.

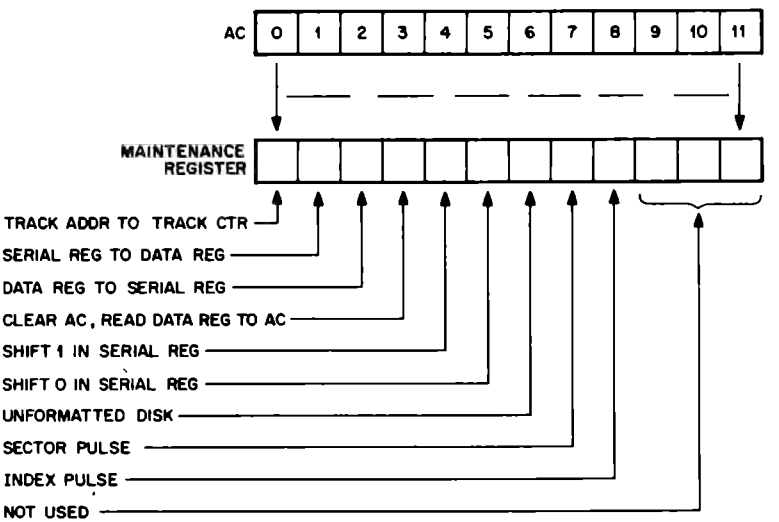


Figure 7-6 Maintenance Word

Skip on Disk Done (DSKD)

Octal Code: 6745
Execution Time: 3.6 μs
Operation: The Disk Transfer Done Flag is sensed; if it is set, the next instruction is skipped.

Skip on Disk Error (DSKE)

Octal Code: 6747
Execution Time: 4.6 μs
Operation: The Error Flag is sensed; if it is set, the next instruction is skipped.

Clear All (DCLA)

Octal Code:	6751
Execution Time:	2.6 μs
Operation:	Clears selected disk to track 000, and clears all control registers and flags except Disk Selection. Transfer Done is set when disk is positioned to track 000.

Read Word Count Register (DRWC)

Octal Code:	6752
Execution Time:	2.6 μs
Operation:	Clears the AC, then reads contents of WC register to AC0-11.

Load Word Count Register (DLWC)

Octal Code:	6753
Execution Time:	3.6 μs
Operation:	Loads the contents of AC into WC register, then clears AC.

Load Current Address Register (DLCA)

Octal Code:	6755
Execution Time:	3.6 μs
Operation:	Loads the contents of AC into CA register, then clears AC.

Read Current Address Register (DRCA)

Octal Code:	6757
Execution Time:	4.6 μs
Operation:	Clears the AC, then reads contents of the CA register to AC0-11.

Example Subroutine

```
DISKIO,  0              /SUBROUTINE ENTRY POINT
         TAD CNT        /FETCH # OF WORDS TO BE READ
         DLWC           /LOAD WORD COUNT REGISTER
         TAD CUR        /FETCH ADD OF MEMORY BUFFER
         DLCA           /LOAD CUR ADDR REGISTER
         TAD 0000       /PUT COMMAND IN ACCUMULATOR
         DLDC           /LOAD COMMAND REGISTER
         TAD ADR        /FETCH TRACK, SURFACE, & SECTOR
         DLDR           /LOAD DISK ADR AND READ DATA
                        /DLDW WOULD HAVE WRITTEN
         DSKD           /DONE WITH TRANSFER?
         JMP.—1         /NO—GO BACK 1
         DSKE           /YES—IS THERE AN ERROR?
         JMP I DISKIO   /NO ERROR—RETURN
         JMP ERR        /GO TO ERROR SUBROUTINE
CNT,     XXX            /# WORDS TO BE READ
CUR,     YYY            /BEG ADDR OF MEMORY BUFFER
ADR,     ZZZ            /DISK ADDRESS
```

DF32-D DEC Disk File & Control & DS32-D DEC Disk File Expander

The DF32-D Disk File is a fast, low-cost, random-access, bulk-storage device and control for use with the PDP-8/E computer. [When the DF32-D is used with the PDP-8/E, the KD8-E Data Break interface and the KA8-E Positive I/O Bus interface are also required.] Operating through the three-cycle Data Break Facility, the DF32-D provides 32,768 13-bit words (12 bits plus parity) of storage, and is economically expandable to 131,072 words when using the DS32-D Expander Disk.

Transfer rate of the DF32-D is 32 or 64 μs per word (optional when timing track is written); average access time is 16.67 ms for 60 Hz power (20 ms with 50 Hz power).

Two basic assemblies make up the DF32-D; the storage unit with read/write electronics and computer interface logic. The storage unit contains a nickel-cobalt-plated disk, driven by a hysteresis synchronous motor. Data is recorded on a single disk surface by 16 fixed-position read/write heads.

Disk motor and shaft, read/write data heads, and timing and address heads are mounted on a 19-inch relay rack assembly, which permits easy access to the unit by sliding the unit in and out of a standard Digital Equipment Corporation cabinet.

The DS32-D Extender Disk File is also a slide-mounted assembly with a storage element and read/write electronics. Information transfers are made via the DF32-D logic, and are controlled by the DF32-D.

Specifications

Storage Capacity	32,768 13-bit words; expandble to 131,072 words in increments of 32,768 words, using DS32-D.	
Data transfer rate	60 Hz power 32 (64) μs per word	50 Hz power 39 (78) μs per word
Average access time	16.67 ms	20.0 ms
Write lock switches	Inhibit writing on lower and/or upper 16K or any 32K disk surface; may be used to inhibit one or more 32K disks in an expanded configuration.	
Addressing Scheme	Random or absolute addressing from 0 to 32K words with variable block sizes from 1 word to 4096 words.	
Data assembly	Read/write on disk is serial, with external transfer parallel by word.	
Data Availability	16 μs (48 μs with alternate timing track) from the time word is assembled until new word starts to shift into assembly register. (A similar timing condition exists during the write operation.)	
Data tracks	16 per disk, 2048 words per track	
Recording method	NRZI	
Density (max)	1100 BPI	
Timing tracks	2 plus 2 spare	

Size	10-½ in. high and 23-⅝ in. deep in a standard 10-in. rack.	
Heat Dissipation	1700 Btu/hr.	
Data Transfer Path	3-cycle Break	Address Locations 7750 Word Count 7751 Memory Address
Program Interrupt	Data Transfer-completion flag and/or non-existent disk.	
Write lock Switches	Inhibit write only on lower or upper 16K or both on one or more discs.	
Select Switches	Rotary Switches to select disk unit number.	

Programming

The following instructions operate the disk system:

Clear Disk Memory Address Register (DCMA)

Octal Code:	6601
Execution Time:	2.6 μs
Operation:	Clears disk memory address register, parity error, and completion flags. This instruction also clears the disk memory request flag and interrupt flags.

Load Disk Memory Address Register and Read (DMAR)

Octal Code:	6603
Execution Time:	3.6 μs
Operation:	Loads the content of the AC into the disk memory address register an clears the AC. This IOT initiates readings of information from the disk into the specified core location. Clears parity error and completion flags.

Load Disk Memory Address Register and Write (DMAW)

Octal Code:	6605
Execution Time:	3.6 μs
Operation:	Loads the content of the AC into the disk memory address register and clears the AC. This disk then begins to write information into the disk from the specified core location. Clears parity error and completion flags. Data break must be allowed to occur within 33 μs (66 μs) after issuing this instruction

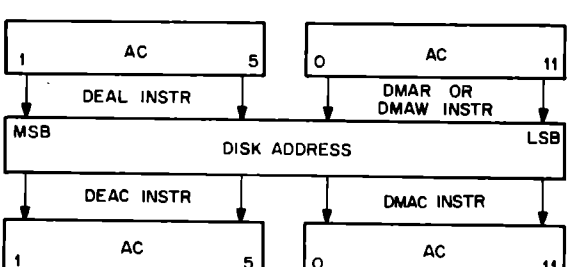


Figure 7-7 Address Word (write)

Clear Disk Extended Address Register (DCEA)

Octal Code: 6611
Execution Time: 2.6 μs
Operation: Clears the disk extended address and memory address extension register.

Skip on Address Confirmed Flag (DSAC)

Octal Code: 6612
Execution Time: 2.6 μs
Operation: Skips next instruction if address confirmed flag is a one. Flag is set for 16 μs whenever the address on the disk equals the contents of the disk address registers. Clears the AC.

Load Disk Extended Address (DEAL)

Octal Code: 6615
Execution Time: 3.6 μs
Operation: Clears the disk extended address and memory address extension registers and loads them with the track address data held in the AC. ORs the contents of these registers, plus the photocell mark and three error flags, into the AC. (See DEAC instruction.)

Read Disk Extended Address Register (DEAC)

Octal Code: 6616
Execution Time: 3.6 μs
Operation: Clears the AC, then loads the contents of the disk extended address register into the AC to allow program evaluation. Skips the next instruction if address confirmed flag is a one.

NOTE

Write lock switch status is true only when disk unit contains a write command. The nonexistent disk condition will appear following the completion of a data transfer during read, where the address acknowledged was the last address of a disk and the next word to be addressed falls within a nonexistent disk. The completion flag for this data transfer is set by the nonexistent disk condition 16 μs after the data transfer.

Skip On Zero Error Flag (DFSE)

Octal Code: 6621
Execution Time: 2.6 μs
Operation: Skips the next instruction if parity error, data request late, and write lock switch flag are all zero. Indicates no errors.

Skip on Data Completion Flag (DFSC)

Octal Code: 6622
Execution Time: 2.6 μs
Operation: Skips the next instruction if the completion flag is a one, indicating data transfer is complete.

Read Disk Memory Address Register (DMAC)

Octal Code: 6626
Execution Time: 3.6 μs
Operation: Clears the AC, then loads the contents of the disk memory address dress register into the AC to allow program evaluation. During read, the final address will be the last one transferred.

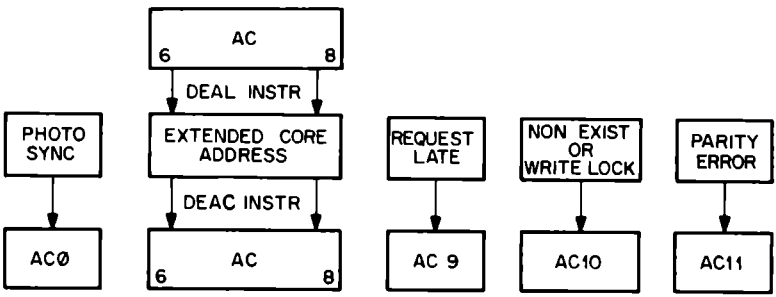


Figure 7-8

Three-cycle data break locations: Work Count address is 7750 (field 0). Current Address is 7751 (field 0).

Three maintenance IOTs are also used by the DF32-D. These IOTs are used to simulate certain pulses within the disk control for static logic tests. Since they all use device code 63, this code should not be used by other peripheral devices when a DF32-D is part of the system.

NOTE

For the DEAL and DEAC instructions, refer to the diagrams shown below:

Bits 1-5 (DEAL, DEAC Inst.)	Accumulator + (Low Order 12 Bits) 0-11 of DMAW or DMAR	→	Disk Address (17 Bit)
Field Bits 6-8 (DEAL, DEAC Inst.)	+ Cell 7751 (Current Address)	→	Current Address (Memory) Address (15 Bit)

The computer can handle 12 bits; therefore, the high order bits for disk and memory address are manipulated by the DEAL and DEAC instructions. Low order bits are manipulated in the AC.

The disk address is a 17-bit value. Bit 1 of the DEAL and DEAC instructions is the most significant bit. The memory address is a 15-bit value. Bit 6 of the DEAL and DEAC instructions is the most significant bit.

Note that the word count 7750 is loaded with the two's complement of the number of words to be transferred, and that the disk address is loaded with the desired starting address. The memory current address (7751) is loaded with the desired address minus one.

MAGNETIC TAPE OPTIONS

DEC tape

The DECtape system is a standard option for the PDP-8/E that serves as an auxiliary magnetic tape data storage facility. The DECtape system stores information at fixed positions on magnetic tape, as in magnetic disk or drum storage devices, rather than at unknown or variable positions, as in conventional magnetic tape systems. This feature allows replacement of blocks of data on tape in a random fashion without disturbing other previously recorded information. In particular, during the writing of information on tape, the system reads format (mark) and timing information from the tape and uses this information to determine the exact position at which to record the information to be written. Similarly, in reading, the same mark and timing information has a number of features to improve its reliability and make it exceptionally useful for program updating and program editing applications. These features are: phase or polarity sensed recording on redundant tracks, bidirectional reading and writing, and a simple mechanical mechanism utilizing hydrodynamically lubricated tape guiding (the tape floats on air over the tape guides while in motion).

Magnetic tape options operated on the external bus of the PDP-8/E require the use of the KA8-E Positive I/O Bus Interface module and the KD8-E Data Break Interface module as prerequisites.

DECtape Format

DECtape utilizes a 10-track read/write head. Tracks are arranged in five nonadjacent redundant channels: a timing channel, a mark channel, and three information channels. Redundant recording of each character bit on nonadjacent tracks materially reduces bit dropouts and minimizes the effect of skew. The series-connection of corresponding track heads within a channel and the use of Manchester phase recording techniques, rather than amplitude sensing techniques, virtually eliminate dropouts.

The timing and mark channels control the timing of operations within the control unit and establish the format of data contained on the information channels. The timing and mark channels are recorded prior to all normal data reading and writing on the information channels. The timing of operations performed by the tape drive and some control functions are determined by the information on the timing channel. Therefore, wide variations in the speed of tape motion do not affect system performance. Information read from the mark channel is used during reading and writing data to indicate the beginning and end of data blocks and to determine the functions performed by the system in each control mode. During normal data reading, the control assembles 12-bit computer-length words from four successive lines read from the information chan-

nels of the tape. During normal data writing, the control disassembles 12-bit words and distributes the bits so they are recorded on four successive lines on the information channels. A mark-channel error-check circuit ensures that one of the permissible marks is read in every six lines on the tape. This 6-line mark-channel sensing requires that data be recorded in 12-line segments (12 being the lowest common multiple of 6-line marks and 4-line data words) which correspond to three 12-bit words.

A tape contains a series of data blocks that can be of any length which is a multiple of three 12-bit words. Block length is determined by information on the mark channel. A uniform block length is usually established over the entire length of a reel of tape by a program that writes mark and timing information at specific locations. The ability to write variable-length blocks is useful for certain data formats. For example, small blocks containing index or tag information can be alternated with large blocks of data. (Software supplied with DECtape allows writing for fixed block lengths only.)

Between the blocks of data are areas called interblock zones. The interblock zones consist of 30 lines on tape before and after a block of data. Each of these 30 lines is divided into five 6-line control words. These 6-line control words allow compatibility between DECtape written on any of DEC's 12-, 18-, or 36-bit computers. As used on the PDP-8/E, only the last four lines of each control word are used.

Block numbers normally occur in sequence from 1 to n. There is one block numbered 0 and one block n + 1. Programs are entered with a statement of the first block number to be used and the total number of blocks to be read or written. The total length of the tape is equivalent to 849,036 lines, which can be divided into any number of blocks up to 4096 by prerecording of the mark track. The maximum number of blocks is determined by the following equation in which n(b) equals number of blocks and n(w) equals number of words per block (n(w) must be divisible by 3).

$$n(b) = \frac{212112}{n(w) + 15} - 2$$

DECtape format is illustrated in Figures 7-9 through 7-12.

Figure 7-9 DECtape Track Allocations

Figure 7-10 DECtape Mark Channel Format

Forward direction of tape motion
ONE BLOCK 86_{10} 18-BIT WORD LOCATIONS
129_{10} 12-BIT DATA WORD LOCATIONS
EXPAND CODE
BLOCK MARK
REVERSE GUARD
LOCK
REVERSE CHECK SUM
REVERSE PRE-FINAL
DATA
DATA
DATA
DATA
DATA
PRE-FINAL
CHECK SUM
REVERSE LOCK
GUARD
REVERSE BLOCK MARK
EXPAND CODE
PERMITS EXPANSION OF BLOCK NUMBER
SIGNIFIES START OF BLOCK AND ALLOWS COMPUTER PROGRAM TO IDENTIFY BLOCKS
PROVIDES WRITE PROTECTION IN REVERSE DIRECTION AND SYMMETRY
NOT USED IN PDP-8 CONFIGURATIONS BUT PROVIDED TO ALLOW COMPATIBILITY WITH OTHER COMPUTERS
PROVIDES SYMMETRICAL ERROR PROTECTION IN BOTH DIRECTIONS
SAME AS FINAL IN REVERSE DIRECTION (FIRST DATA WORD)
SAME AS PRE-FINAL IN REVERSE DIRECTION (SECOND DATA WORD)
SUCCESSIVE DATA WORDS
PERMITS EXPANSION OF BLOCK NUMBER
BLOCK NUMBER AS REVERSE DIRECTION
SAME FUNCTION AS REVERSE LOCK
PROTECTS TAPE IN EVENT OF MARK CHANNEL ERRORS
PROVIDES AUTOMATIC ERROR DETECTION AND END OF BLOCK DETECTION WHEN READING
IDENTIFIES FINAL DATA WORD AND REQUESTS CHECKSUM
NOT USED BUT PROVIDED TO ALLOW TAPE COMPATIBILITY WITH OTHER SYSTEMS
ADDITIONAL DATA WORDS
NOTE: END MARKS WHICH IDENTIFY THE PHYSICAL ENDS OF THE TAPE, ARE THE ONLY MARKS NOT SHOWN
Code functions listed apply only in the direction indicated.

Figure 7-11 DECtape Control Word and Data Word Assignments

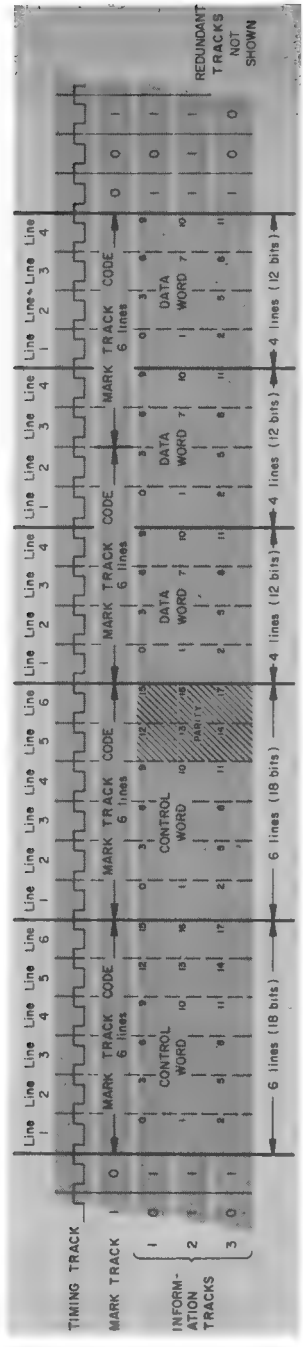


Figure 7-12 DECtape Format Details

TU56 Dual DECtape Transport and TC08-P DECtape Control

A DECtape system on the external bus can contain up to four TU56 Dual DECtape transports (the equivalent of up to eight single tape transports) controlled from one TC08-P unit. Data transfers between the computer and tape are implemented using the three-cycle data break facilities of the computer (refer to Chapter 10 for three-cycle data break description). Thus, the KA8-E Positive I/O Bus Interface and KD8-E Data Break Interface units are prerequisites.

Data is stored on tape in the form of three-bit words (refer to tape format) and is transferred between the tape and computer in the form of 12-bit words. A 12-bit read/write buffer in the TC08 assembles and disassembles the information for transfer. For transfers to the computer, data is read from four consecutive lines of tape and assembled into a 12-bit word. When transferred to the computer, the 12-bit word is supplied via external bus lines DATA00-11 to the KD8-E Data Break Interface. This unit, in turn, provides the word to OMNIBUS lines DATA0-11 under data break control. For transfers to tape, the KA8-E unit buffers the 12-bit words and provides them to the TC08 via external bus lines BMB00-11. The TC08 disassembles these words and supplies them to the tape transport for the writing of four tape lines. Transfer of command and control signals is effected by IOT instructions. These instructions are provided to the TC08 via the BMB00-11 external bus lines.

The TC08 contains registers and control flip-flops that form two status registers (designated A and B) for transfer of information to and from the computer accumulator.

TU56 Dual DECtape Transport

The TU56 provides the PDP-8/E user with a compact, high-reliability dual-reel tape transport in just 10½ inches of rack space. When used with the TC08-P control, the TU56 provides two fixed -address, magnetic tape facilities for high-speed loading, readout, and updating of programs and data. The TU56 transport contains the tape read/write heads, drive mechanisms, and switching circuits for tape drive and direction. All transport operations (except local) are controlled by the TC08 from program instructions. The TC08 selects the transport, controls tape motion and direction, selects a read or write operation and buffers data transferred. Information is stored in the form of three-bit words on a one-mil Mylar tape with ten tracks. This tape, ¾ inches in width and 260 feet in length, is contained on a reel that is less than four inches in diameter. Information can be recorded or read for either direction of tape motion.

Redundant recording (each bit of data and timing is recorded on two tracks) ensures high reliability and eliminates the need for parity checking. Data words are recorded on six of the ten tracks and four tracks are allotted for mark and timing channels. Other features include TTL logic, dynamic braking for shorter turnaround time, and DC motor drive to eliminate line frequency dependency. Connections from the read/write head are made directly to the external control, which contains the read and write amplifiers.

The logic circuits of the TU56 transport control tape movement in either direction over the read/write head. Tape drive motor control is com-

pletely through the use of solid-state switching circuits to provide fast, reliable operation. These circuits control the torque of the two motors that transport the tape across the head according to the established function of the device: i.e., go, stop, forward, or reverse. In normal tape movement, full torque is applied to the forward or leading motor and a reduced torque is applied to the reverse or trailing motor to keep proper tension on the tape. Since tape motion is bidirectional, each motor serves as either the leading or trailing drive for the tape, depending upon the forward or reverse control status of the TU56.

Tape movement can be controlled by commands originating in the computer or by manual operation of switches on the front panel of the transport. Manual control is used to mount new reels of tape on the transport, or as a quick maintenance check for proper operation of the control logic in moving the tape.

Since DECtape is a fixed address system, the programmer need not know accurately where the tape has stopped. To locate a specific point on tape he must only start the tape motion in the search mode. The address of the block currently passing over the head will be automatically transferred to core where it can be compared with the desired block address and tape motion continued or reversed accordingly. TU56 typical time characteristics are provided below, but are not accurately controlled.

Start Time	150 ms*
Stop Time	100 ms*
Turnaround Time	200 ms*

*Also, see control specifications. These times are frequently lengthened by the particular control.

Specifications

Transfer rate	33,300 three-bit characters per second
Information capacity	2.7 million bits per reel
Density	350 + or − 55 bits per inch
Tape speed	93 + or − 12 inches per second
Tape motion	Bidirectional
Start time	150 + or − 15 ms
Stop time	100 + or − 10 ms
Turn around time	200 + or − 50 ms
Reel capacity	250 ft. of ¾ inch, 1 mil Mylar tape
Reel size	3.9 inches in diameter
Mounting	Mounts in a standard 19-inch equipment rack
Size	10½ in. high, 19 in. wide, 9¾ in. deep
Cooling	Internally mounted fans provided
Power requirements	a. + 10V @ 0.53 amps or + 5V @ 0.55 amps b. − 15V @ 0.45 amps c. 115/220 VAC + or − 10% @ 2.85/1.43 amps 47-63 Hz
Environmental	Temperature: 40 degrees F to 90 degrees F Humidity: 15% to 80% Relative Humidity Internal Temp Rise: 10% F above ambient
Reliability	Recoverable Error Rate-less than 1 part in 2.5 x 10 ↑ 10 transfers

TC08-P DECtape Control

The TC08-P control buffers and controls information transfers between one to eight TU56 transports (one to four TU56 Dual DECtape transports) interfacing with the external bus of the PDP-8/E. Transfers are implemented using the three-cycle data break facilities of the computer; thus, the KA8-E Positive I/O Bus Interface and the KD8-E Data Break Interface modules are prerequisites.

During both input and output operations, the TC08-P receives data and control information from the processor and generates the appropriate signals to the selected transport to execute the programmed commands. Binary information is transferred between the tape transport and the computer as one 12-bit computer word every 133⅓ μs. When writing, the TC08-P disassembles the 12-bit word into four successive three-bit words to be written on tape. During read operations, the TC08-P assembles the four successive three-bit words into one 12-bit word for transfer to the computer. Transfers between the computer and the control always occur in parallel for a 12-bit word. Data transfers use the three-cycle data-break (high speed channel) facility of the computer. (Refer to Chapter 10 for details of 3-cycle data-break transfers.)

The TC08-P contains the following primary control and data processing circuits:

a. Device selector and IOT decoding logic to command a transport from program instructions.
b. A 12-bit buffer register for assembling tape inputs and disassembling computer data.
c. A command and status register (designated Status Register A) for defining: (1) the active transport, (2) direction of tape, (3) tape motion, (4) operating mode, (5) function (read/write, search, etc.), (6) interrupt enable, and (7) clearing of flags.
d. A status register (designated Status Register B) for indicating error status and other status.
e. Flag circuits that provide the program with conditional indications and requests.
f. Tape motion and direction control circuits.
g. Mark track generation and detection circuits with error detectors.
h. Longitudinal parity generation and checking circuits.
i. Data break request circuits.

Programmed IOT instructions are generated to clear, read, or load Status Register A and to read or load Status Register B. An IOT skip instruction is also provided to test the status of flag circuits. These instructions are provided to the TC08-P control via the KA8-E Positive I/O Bus Interface and external bus lines BMB-00-11.

A control and indicator panel is also provided with the TC08-P. A single control, NORMAL/WRTM, places the TC08-P in the write timing and mark track mode (WRTM), or else in the NORMAL mode. The indicators denote the current status of the control including the tape transport selected, motion, function, interrupt status, error flags, and other status indications.

Three program flags in the TC08-P control serve as condition indicators and request originators.

a. DECtape Flag (DT): This flag indicates the active/done status of the current function.
b. Data Flag (DF): This flag requests a data break to transfer a block number into the computer during a search function, or when a data word transfer is required during a read or write function.
c. Error Flag (EF): Detection of any nonoperative condition by the control sets this flag in status register B and stops (except for parity errors) the selected transport. The error conditions indicated by this flag are:
   (1) Mark Track Error: This error occurs any time the information read from the mark channel is erroneously decoded.
   (2) End of Tape: The end zone on either end of the tape is over the read head.
   (3) Select Error: This error occurs 5 μs after loading status register A to indicate any one of the following conditions:
   (a) Specifying a unit select code which does not correspond to any transport select number, or which is set to multiple transports.
   (b) Specifying a write function with the WRITE ENABLED/WRITE LOCK switch in the WRITE LOCK position on the selected transport.
   (c) Specifying an unused function code (i.e., AC6-8 = 111).
   (d) Specifying any function except write timing and mark track with the NORMAL/WRTM switch in the WRTM position.
   (e) Specifying the write timing and mark track function with the NORMAL/WRTM switch in the NORMAL position.
   (4) Parity Error: This error occurs during a read data function if the longitudinal parity or check sum over the entire data word, the reverse check character, and the check character is not equal to 1.
   (5) Timing Error: This error indicates a program fault caused by one of the following conditions:
   (a) A data break did not occur within 17 μs (+ or − 30%) of the data break request.
   (b) The DT flag was not cleared by the program before the control attempt to set it.
   (c) The read data or write data function was specified while a data block was passing the read head.

Three-cycle data break locations: The TC08-P uses location 7754 of field 0 for word count and 7755 of field 0 for current address.

Control Modes—The DECtape system operates in either the normal or continuous mode, as determined by bit 5 of status register A during a DTXA command. Operation in each mode is as follows:

a. Normal (NM): Data transfers and flag settings are controlled by the format of information on the tape.

b. Continuous (CM): Data transfers and flag settings are controlled by a word count read from core memory during the first cycle of each three-cycle data break, and by tape format.

Functions—The DECtape system performs one of seven functions, as determined by the octal digit loaded into status register A during a DTXA command. These functions are:

a. Move: Initiates movement of the selected transport tape in either direction. Mark channel decoding is inhibited in this mode except for end of tape.
b. Search: As the tape is moved in either direction, sensing of a block mark causes a data transfer of the block number. If the word count overflows in either NM or CM, the DT flag is set and causes a program interrupt. After finding the first block number, the CM can be used to avoid all intermediate interrupts between the current and the desired block number. This makes a virtually automatic search possible.
c. Read Data: This function is used to transfer blocks of data into core memory with the transfer controlled by the tape format. In NM, the DT flag is set at the end of a block and causes a program interrupt. In CM, transfers stop when the word count overflows, the remainder of the block is read for parity checking, and then the DT flag is set.
d. Read All: Read all is used to read tape in an unusual format, since it causes all lines to be read. In NM, the DT flag is set at each data transfer. In CM, the DT flag is set when WCO occurs. In either case, the DT flag causes a program interrupt.
e. Write Data: This function is used to write blocks of data with the transfer controlled by the standard tape format. After word count overflow occurs, zeros are written in all lines of the tape to the end of the current block. Then the parity checksum for the block is written. The DT flag rises as in the read function.
f. Write All: The write all function is used to write an unusual tape format (e.g., block numbers). The DT flag assertions are similar to the read all function.
g. Write Timing and Mark Track: This function is used to write on the timing and mark tracks, permitting blocks to be established or block lengths to be changed. The DT flag assertions are also similar to the read all function. This function is illegal unless a manual switch in the control is positioned to WRTM.

Programmed Operation—Prerecording of a reel of DECtape, prior to its use for data storage, is accomplished in two passes. During the first pass, the timing and mark channels are placed on the tape. During the second pass, forward and reverse block mark numbers, the standard data pattern, and the automatic parity checks are written. These functions are performed by the DECTOG program. Prerecording utilizes the write timing and mark channel function, and a manual switch on the control, which permits writing on the timing and mark channels, activates a clock, which produces the timing channel recording pattern and enables flags for program control. Unless this control function and switch are used simultaneously, it is physically impossible to write on the

mark or timing channels. An indicator lamp on the control panel lights when the manual NORMAL/WRTM switch is in the WRTM position. Under these conditions only, the write register and write amplifier, used to write on information channel 1 (bits 0, 3, 6, and 9), are used to write on the mark channel. This prerecording operation need only be performed once for each reel of DECtape.

There are two registers in the TC08 DECtape Control that govern tape operation and provide status information to the operating program. Status register A contains three unit selection bits, two motion bits, the continuous mode/normal mode bit, three function bits, and three bits that control the flags. Status register B contains the three memory field bits and the error status bits. PDP-8/E IOT microinstructions are used to clear, read, and load these registers. In addition; there is an IOT skip instruction to test control status.

Since all data transfers between DECtape and the computer memory are controlled by the data break facility, the program must set the WC and CA registers (locations 7754 and 7755, respectively) before a data break. After initiating a DECtape operation, the program should always check for error conditions (a program interrupt would be initiated if the error flag is enabled and if the program interrupt system is enabled). The DECtape system should be started in the search function to locate the block number selected for transfer; when the correct block is found, the transfer is accomplished by programmed setting of the WC, CA, and status register A.

When searching, the DECtape control reads block numbers only. These are used by the operating program to locate the correct block number. In NM, the DECtape flag is raised at each block number. In CM, the DECtape flag is raised only after the word count reaches zero. The current address is not incremented during searching and the block number is placed in core memory at the location specified by the content of the CA. Data is transferred to or from the computer core memory from locations specified by the CA register which is incremented by one before each transfer.

Each time the DECtape system is ready to transfer a 12-bit word, and when the start of the data position of the block is detected, the data flag is raised to initiate a data break request to the data break facility. Therefore, the main computer program continues running, but is interrupted approximately every 133⅓ μs for a data break to transfer a word. Transfers occur between DECtape and successive core memory locations specified by the CA. The initial transfer address minus one is stored in the CA by an initializing routine. The number of words transferred is determined by the tape format in NM, or by tape format and the word count in CM. At the conclusion of the data transfer, the DT flag is raised and a program interrupt occurs. The interrupt subroutine checks the DECtape error bits to determine the validity of the transfer, and either initiates a search for the next information to be transferred or returns to the main program.

During all normal writing transfers, a check character (the six-bit logical equivalent of the words in the data block) is computed automatically by

the control and is recorded automatically as one of the control words immediately following the data portion of the block. This same character is used during reading to determine that the data playback and recognition take place without error.

Any one of the eight tape transports may be selected for use by the program. After using a particular transport, the program can stop the transport currently being used and select another transport, or can select another transport while permitting the original selection to continue running. This is a particularly useful feature when rapid searching is desired, since several transports may be used simultaneously. Caution must be exercised, however; although the original transport continues to run, no tape-end detection or other sensing takes place. Automatic tape-end sensing that stops tape motion occurs in all functions, but only in the selected tape transport.

The following is a list of timing considerations for programmed operations. (These times are based on 129 12-bit data words per block.)

n(s) = the number of block numbers to be read in the search function and CM, counting through the one causing the word count overflow. Only the block number causing the word count overflow requests a program interrupt.

n(d) = number of words transferred divided by the number of words per block. If the remainder does not equal 0, use the next larger whole number.

n(A) = number of words transferred.

OPERATION	TIMING
Answer a data break request	Up to 17 μs + or − 30%
Word transfer rate	One 12-bit word every 133 μs + or − 30%
Block transfer rate	One 129-word block every 18.2 ms + or − 30%
Change funtcion from search to read data for the current block after DT flag from block number	400 μs + or − 30%
Change function from search to write data for current block after DT flag from block number	400 μs + or − 30%
Change function from read data to search for the next block after DT flag from transfer completion	1000 μs + or − 30%
Change function from write data to search for next block after DT flag from transfer completion	1000 μs + or − 30%
DECtape flag rises in continuous mode	
Move function	Never
Search function	(n(s)) x (18.2 ms + or − 30%)
Read data function	(n(D)) x (18.2 ms + or − 30%)
Read all function	(n(A)) x (133 μs + or − 30%)

OPERATION	TIMING
Write data function	(n(D)) x (18.2 ms + or − 30%)
Write all function	(n(A)) x (133 μs + or − 30%)
Write T & M function	(n(A)) x (133 μs + or − 30%)
In normal mode	
Move function	Never
Search function	Every 18.2 ms + or − 30%
Read data function	Every 18.2 ms + or − 30%
Read all function	Every 133 μs + or − 30%
Write data function	Every 18.2 ms + or − 30%
Write all function	Every 133 μs + or − 30%
Write T & M function	Every 133 μs + or − 30%

Programming

The following instructions are associated with TC08-P operation:

Read Status Register A (DTRA)

Octal Code: 6761
Execution Time: 2.6 μs
Operation: Transfers content of Status Register A to the AC. ORs AC0-9 with Status Register with the result appearing in AC. The AC is not cleared before the transfer. AC bit assignments are defined in Figure 7-13.

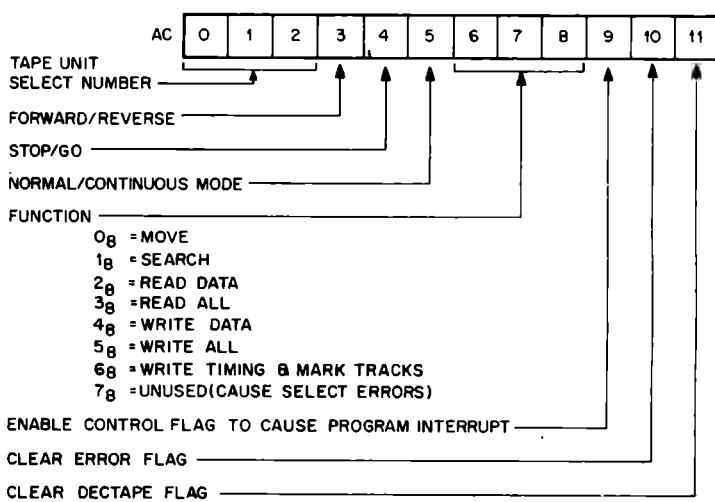


Figure 7-13 Status Register A Bit Assignments

Clear Status Register A (DTCA)

Octal Code: 6762
Execution Time: 2.6 μs
Operation: Clears Status Register A; DECtape and Error flag are undisturbed.

Clear and Load Status Register A (DTLA)

Octal Code: 6766
Execution Time: 3.6 μs
Operation: Clears Status Register A, then EXCLUSIVE ORs content of AC0-9 into Status Register A. Samples AC10 and 11 to control clearing of DECtape and error flags, then clears AC.

Load Status Register A (DTXA)

Octal Code: 6764
Execution Time: 2.6 μs
Operation: EXCLUSIVE ORs content of AC0-9 into Status Register A. Samples AC bits 10 and 11 to control clearing of Error and DECtape flags, then clears the AC.

Skip On Flag (DTSF)

Octal Code: 6771
Execution Time: 2.6 μs
Operation: If either DECtape or Error flags is set, skips the next instruction.

Read Status Register B

Octal Code: 6772
Execution Time: 2.6 μs
Operation: ORs content of Status Register B into AC. The AC is not cleared before transfer; AC bit assignments are defined in Figure 7-14.

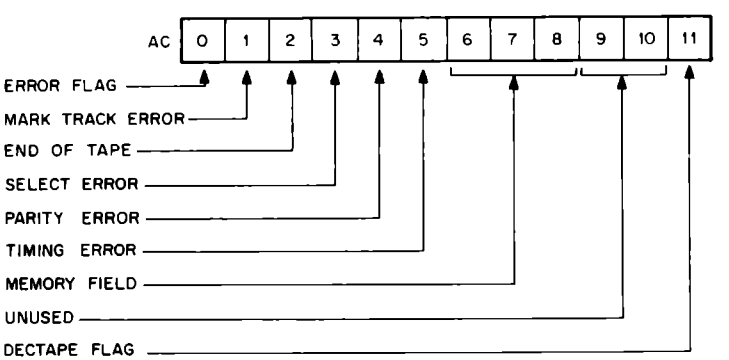


Figure 7-14 Status Register B Bit Assignments

Load Status Register B

Octal Code: 6774
Execution Time: 2.6 μs
Operation: Loads memory field portion of Status Register B with content of AC6-8, then clears the AC.

An elementary subroutine for reading or writing DECtape is given below. This routine does not use the interrupt, and exits with the DECtape drive halted.

The format for calling the subroutine is:

```
JMS (IDTAPE)    Effective JMS to IDTAPE, i.e., indirect JMS if IDTAPE
                  is not on same page as calling sequence.
WORD 1,         Bits 0-2, unit number
                Bit 3, start search (0=forward 1=reverse)
                Bits 6-8, memory field for transfer
                Bit 10, error return (0=JMP WORD 5)
                                     (1=JMP I WORD 5)
                Bit 11, function (0=READ 1=WRITE)
WORD 2,         Block number for start of transfer
WORD 3,         2's complement of the number of words to transfer
WORD 4,         Memory address of first transfer minus 1
WORD 5,         Error return or address for error return (to correspond
                  to Bit 10 of Word 1)
RETURN,         Transfer completed, return with AC cleared
```

```
ID7400,  7400
                                /AND MASK (MUST BE FIRST CELL IN
                                /PAGE)
IDTAPE,  0                      /ENTRY TO SUBROUTINE
         CLA
         TAD     I IDTAPE       /SAVE WORD 1
         DCA     IDCODE
         ISZ     IDTAPE         /ADVANCE TO BLOCK NUMBER (WORD
                                /2)
         TAD     IDCODE
ID0200,  AND     ID7400         /UNIT NUMBER AND DIRECTION BIT
         TAD     ID0010         /PUT INTO SEARCH MODE
         DTCA    DTXA
         DTLB                   /CLEAR FIELD BITS
         TAD     IDWC
         DCA     I IDCA         /SET UP CURRENT ADDRESS (7755)

/ERROR WHILE SEARCHING . . . NORMALLY ENTERED WITH B
/STATUS REGISTER IN THE AC, PERFORMS TURN AROUND IF END
/ZONE ERROR, AND FORCES THE STOP-GO BIT TO GO

IDSERR,  RTL
         RAL                    /MOVE END ZONE FLAG TO LINK
         CLA     CML
         TAD     ID0200         /GET DECTAPE GO FLAG

/CHANGE DIRECTION IF AND ONLY IF THE LINK IS ZERO,

IDCONT,  SNL                    /CHECK DIRECTION AND SIGN
         TAD     ID0400         /REVERSE DIRECTION
         DTXA                   /ENTER AND GO IN SEARCH MODE
         DTSF    DTRB           /IDLE . . AND LOAD ERROR FLAG
         JMP     .-1            /WAIT UNTIL FLAG COMES UP
         SPA                    /TEST ERROR FLAG
         JMP     IDSERR
```

```
        DTRA                    /GET DIRECTION BIT
        RTL
        RTL                     /DIRECTION BIT GOES TO LINK
        SZL     CLA
        TAD     ID0002          /REVERSE . . . GET "BLOCK TO FIND"
                                /-2
        TAD     I IDWC          /ADD IN LAST BLOCK SEEN
        CMA                     /COMPLEMENT
        TAD     I IDTAPE        /ADD IN "BLOCK TO FIND"
        CMA
        SZA     CLA             /BLOCK NUMBERS MATCH?
        JMP     IDCONT          /REENTER SEARCH LOOP
        SZL                     /CHECK DIRECTION BIT
        JMP     IDCONT+1        /TURN AROUND IF REVERSE

/END OF SEARCH LOOP, TAPE IS NOW AT DESIRED BLOCK
/TRAVELING IN A FORWARD DIRECTION,

        ISZ     IDTAPE
        TAD     I IDTAPE        /GET WORD COUNT
        DCA     I IDWC
        ISZ     IDTAPE
        TAD     I IDTAPE        /GET TRANSFER ADDRESS
        DCA     I IDCA
        TAD     IDCODE
        DTLB                    /LOAD FIELD BITS
        IAC                     /GET READ-WRITE FLAG
        AND     IDCODE
        RTL     CLL             /MULTIPLY BY 20 (OCTAL)
        RTL
        TAD     ID0130          /BUILD INSTRUCTION
        DTXA                    /START UP READ OR WRITE
        DTSF    DTRB            /WAIT . . AND LOAD ERROR FLAG
        JMP     .-1
        ISZ     IDTAPE          /ADVANCE TO WORD 5
        SMA                     /SKIP IF ERROR FLAG SET
        ISZ     IDTAPE          /ADVANCE TO WORD 6 . . . NORMAL
                                /EXIT
        SPA     CLA             /SKIP FOR NORMAL EXIT
        TAD     IDCODE          /GET INDIRECT RETURN BIT
        RTR                     /MOVE TO LINK
        SNL     CLA             /SKIP IF JMP I <WORD 5>
        JMP     .+3
        TAD     I IDTAPE        /MAKE DOUBLE INDIRECT RETURN
        DCA     IDTAPE
        DTRA
        AND     ID0200          /GET STOP-GO BIT
        TAD     ID0002          /PRESERVE DECTAPE ERROR FLAGS
        DTXA                    /STOP TAPE
        JMP     I IDTAPE        /EXIT
IDWC,   7754                    /WORD COUNT FOR DATA BREAK
IDCA,   7755                    /CURRENT ADDRESS FOR DATA BREAK
ID0010, 10                      /SEARCH FUNCTION BIT
```

```
ID0400,   400          /FORWARD-REVERSE BIT
ID0130,   130          /USED TO BUILD READ AND WRITE
                       /CODE

ID0002,   2
IDCODE,   0
```

Software

Four types of programs have been developed as DECtape software for the PDP-8/E:

a. Subroutines which the programmer may easily incorporate into a program for data storage, logging, data acquisition, data buffering (queuing), etc.
b. A library calling system for storing named programs on DECtape and a means of calling them with a minimal size loader.
c. System software which provides for storing, assembling, and editing of programs on DECtape, thereby greatly increasing the versatility and flexibility of the PDP-8/E.
d. Programs for preformatting tapes controlled by the content of the switch register to write the timing and mark channels, to write block formats, to exercise the tape and check for errors, and to provide each of maintenance.

Program development has resulted in a series of subroutines which read or write any number of DECtape blocks, read any number of 129-word blocks as 128 words (one memory page), or search for any block (used by read and write, or to position the tape). These programs are assembled with the user's program and are called by a JMS instruction. The program interrupt is used to detect the setting of the DECtape flag, thus allowing the main program to proceed while the DECtape operation is being completed. A program flag is set when the operation has been completed. Thus, the program effectively allows concurrent operation of several input/output devices along with operation of the DECtape system. These programs occupy two memory pages (400 (octal) = 256 (decimal) words).

The library system has the following features: First, the computer state remains unchanged when it exits. Second, the library calls programs by name from the keyboard and allows for expansion of the program file stored on the tape. Finally, the library conforms to existing system conventions, namely, that all of memory except for the last memory page (7600 (octal)—7777 (octal)) is available to the programmer. The PDP-8/E DECtape library system is loaded by a 17 (decimal)—instruction bootstrap routine that starts at address 7600 (octal). This loader calls a larger program into the last memory page, whose function is to preserve on the tape the content of memory from 6000 (octal) through 7577 (octal), and then load the INDEX program and the directory into those same locations. Since the information in this area of memory has been preserved, it can be restored when operations have been completed. The basic system tape contains the following programs:

a. INDEX: Typing this word causes the names of all programs currently on file to be typed out.

b. UPDATE: Allows the user to add a new program to the files. UPDATE queries the operator about the program's name, its starting address, and its location in core memory.

c. GETSYS: Generates a skeleton library tape on a specified DECtape unit.

d. DELETE: Causes a named file to be deleted from the tape.

Starting with the basic library tape, the user can build a complete file of his active programs and continuously update it. One of the uses of the library tape may be illustrated as follows:

The programmer may call the PDP-8/E FORTRAN compiler from the library tape and with it compile the program, obtaining the object program. The FORTRAN operating system may then be called from the library tape and used to load the object program. At this time the library program UPDATE is called, the operator defines a new program file (consisting of the FORTRAN operating system and the object program), adn adds it to the library tape. As a result, the entire operating program and the object program are now available on the DECtape library tape.

The DECtape system software is permanently stored on DECtape, from which it can be rapidly loaded. Any systems programs such as the assemblers (XPAL and XMACRO), the Symbolic Editor (XEDIT), or the Binary Loader (XLOAD), can be loaded in less than one minute.

The system software uses a standard DECtape format. There are 128 (200 (octal)) words per block and 1464 (2701(octal)) blocks, so the user has the remaining 1336 blocks for rapid access storage of his own programs.

The primary advantage for users are:

a. Efficient use of high-speed transfer rates between DECtape and core memory.

b. Symbolic programs may now be stored, edited, and assembled on DECtape, greatly increasing the versatility and flexibility of the PDP-8/E.

c. The computational workload can be more than doubled, compared to high-speed paper tape systems.

User's programs are written exactly as before for assembly by the PAL or MACRO-8 Assemblers. Using the Symbolic Editor, source programs are typed directly onto DECtape. After assembly, fast symbolic debugging can be done with DDT-8—after loading the program symbol table into DDT with the symbolic loader, XSYM.

The Binary Loader (XLOAD) can load the assembled binary program directly from the DECtape for program execution. Source files, symbol table, and program listings can be stored on DECtape and listed later, if desired. A duplicating program, XDUP, is available for copying programs.

This DECtape system also includes system calls to load any program from DECtape, to update or delete source files, and to restore the system for use by another programmer.

Although the system operates with one DECtape, a two DECtape configuration is strongly recommended as it will permit duplication of programs and saving of back-up master tapes. In a single DECtape system, if the system library is accidentally destroyed, the stored data cannot be replaced immediately because there is no means of recovery.

The last group of programs, called DECTOG, is a collection of short routines controlled by the content of the switch register. It provides for the recording of timing and mark channels and permits block formats to be recorded for any block length. Patterns may be written in these blocks and then read and checked. Writing and reading is done in both directions and checked. Specified areas of tape may be "rocked" for specified periods of time. A given reel of tape may thus be thoroughly checked before it is used for data storage. These programs may also be used for maintenance and checkout purposes.

DATA ACQUISITION PERIPHERALS

Digital Equipment Corporation manufactures a number of data acquisition and control subsystems that can be used with the PDP-8/E. In general, these subsystems are modularized so that a small inexpensive subsystem can be purchased initially and expanded as required. Two major analog multiplexer subsystems are available, covering analog input ranges from 10 mv to 100 V and having channel capabilities from 1 to 1024 channels. In addition, two types of Digital-to-Analog Converter subsystems are provided as options. A Universal Digital Controller subsystem is also offered as an option. This subsystem provides capabilities for controlling or interrogating up to 3072 discrete digital loads or sources.

AD01A 12-Bit Analog-to-Digital Converter

The AD01A is a flexible, low-cost, multichannel analog acquisition option for the PDP-8/E computer. The standard AD01A consists of an expandable solid-state multiplexer for four channels, channel selection circuits for up to 32 channels, a programmable input range selector, a 10-μs A/D converter, control, and interface, and power supply mounted on a single module. Module additions can be used to expand the multiplexer in four-channel groups up to 32 channels.

The AD01A interfaces with the external bus of the PDP-8/E computer to provide ten-bit digitization of unipolar analog signals having a full-scale range of 0V to +1.25V, +2.5V, +5V or 10V. An optional sign-bit addition permits eleven-bit bipolar operation. A programmable input range selection extends the dynamic range of the AD01A (at moderate sampling rates) to an equivalent of 13 bits for unipolar inputs or to 14 bits for bipolar inputs. An optional sample and hold amplifier is also available to reduce the conversion aperture to 100 ns. Each multiplexer channel switch utilizes an enhancement mode MOSFET that is normally open when unselected, or when power is off. These switches provide overload protection for up to + or — 20V, and signal protection against short circuits.

Operation of the AD01 is controlled by program instructions from the computer. An ADSC instruction selects the multiplexer channel, system gain, and the interrupt or noninterrupt mode as defined by an AC word.

The selected channel input is connected to the programmable-gain amplifier, which scales the analog input for a 0 to +10V output that is provided to the A/D converter summation junction, either directly or via the sample-and-hold and sign-bit options. The ADSC instruction also clears the A/D Done Flag and initiates a conversion cycle. The conversion is performed by a successive-approximation method requiring approximately 10 μs. For a standard AD01A, the analog input results in a 10-bit binary code. The sign-bit option permits conversion of bipolar inputs (0 to + or − 1.25V, + or − 2.5V, + or − 5V, or + or − 10V) to an eleven-bit, two's complement code with an extended sign format. For this format, AC bits 0 and 1 are connected to the same source and denote polarity of the analog input (1=−V, 0=+V).

When the conversion is complete, the A/D converter sets its A/D Done Flag. This flag is sensed by an ADSF instruction and, if set, causes the next program instruction to be skipped so that the converter word can be transferred to the AC, using an ADRB instruction. The ADRB instruction also clears the A/D Done flag to ready the AD01A for another cycle. The above operations can also be implemented using microprogrammed IOT instructions for "best sampling" operation.

Specifications

Resolution	Unipolar 10 bits, or 1 part in 1024 Bipolar (option) sign + 10 bits
Converter Accuracy	+ or − .05% of 10V full-scale input + or − .075% of full-scale with sample & hold
Quantizing Error	+ or − ½ least significant bit
Thruput Rate	14 μs (70 kHz max) including channel & gain selection 17.5 μs (57 kHz max) with sample & hold
Conversion Amperture	10 μs 0.1 μs with sample & hold
Sample & Hold Acquisition	3.5 μs to .01% of full-scale step change
No. of Analog Inputs	4 minimum, expandable to 32 in groups of 4 Channels
Input Impedance	1000 megohms in parallel with 20 pf.
Input Isolation	Enhancement mode MOSFET switches, "off" when unselected or power off.
Channel Selection (program selectable)	5-bit address
Input voltage range (program selectable)	Unipolar: 0 to + 1.25, + 2.5, + 5.0, +10.0V full-scale Bipolar (option): 0 to + or − 1.25, + or − 2.5V, + or − 5.0, + or − 10.0V full-scale
Overload Capability	+ or − 20V on all ranges without damage

Overload Recovery	Within acquisition time upon removal of overload
Cross Channel Attenuation	78 db, DC-80Hz for 20 volts p-p signals, 100 ohm source impedance
Input Gain	Program selectable

Accuracy Specifications

					Comments
Program Selected Gain	×1	×2	×4	×8	Selectable
Full Scale Input	+10V	+5.0V	+2.5V	+1.25V	Unipolar
Gain Accuracy	0.05%	0.05%	0.05%	0.05%	% of full-scale for 30 days
Resolution	9.8mv	4.9mv	2.45mv	1.22mv	per bit
Zero Drift	550	300	175	110	uv/degreeC RTI
Zero drift w/sample & hold	750	400	225	140	uv/degreeC RTI
Noise	2mv	1mv	1mv	1mv	0-0 RTI
w/sample & hold	+−.05%	1.2mv	1.2mv	1.2mv	0-0
Repeatability	2.4mv	+−.05%	+−.05%	+−.1%	25 degree C and 3 sigma confidence

Sign Bit	Adds one bit without affecting specifications
Word Length	Control word 12 bits Output word 12 bits
Modes of Operation	Interrupting/noninterrupting (program-selectable)
Environmental	0 to + 55 degrees C operating −25 degrees to + 85 degrees C storage to 90% without condensation
Power	115/230V + or − 10% 47 to 470 Hz, single phase less than 50W

Programming

The following instructions are associated with AD01A operation:

Skip on A/D Done Flag (ADSF)

Octal Code:	6531
Execution Time:	2.6 μs
Operation:	Skips the next program instruction if the A/D Done Flag is set.

Read A/D Buffer (ADRB)

Octal Code:	6532
Execution Time:	2.6 μs
Operation:	Transfers the content of the A/D buffer to AC0 through AC11 and clears the A/D Done Flag.

Convert Analog Input (ADCV)

Octal Code:	6534
Execution Time:	2.6 μs
Operation:	Clears the A/D Done Flag and initiates a conversion operation.

Select Multiplexer Channel and Gain (ADSC)

Octal Code: 6535
Execution Time: 3.6 μs
Operation: Loads the content of AC into multiplexer input register and implements channel and gain selection. Also clears the A/D Done Flag and initiates a conversion.

Read A/D Buffer, Clear Flag and Start Conversion (ADRC)

Octal Code: 6536
Execution Time: 3.6 μs
Operation: Jam transfers content of A/D buffer to AC0-AC11, clears flag, and starts a new conversion.

Select Channel and Gain and Read A/D Buffer (ADSR)

Octal Code: 6537
Execution Time: 4.6 μs
Operation: Transfers the content of the AC to the multiplexer input register, clears the AC, and transfers the content of the A/D buffer to the AC. The A/D Done Flag is then cleared, and another conversion is initiated.

NOTE

The ADRC (6536) and ADSR (6537) instructions are used for "burst sampling" conversions in the noninterrupt mode.

A program service routine for the AD01A can be written as follows:

```
ADSC    /SELECT CHANNEL, GAIN AND MODE AND START
        /CONVERSION
ADSF    /SKIP ON A/D DONE FLAG
JMP.—1  /JUMP BACK AND TEST FLAG AGAIN
ADRB    /READ A/D BUFFER
EXIT
```

AFC-8 Low-Level Analog Multiplexer

The AFC-8 Multiplexer is a computer-based unit that multiplexes up to 1024 analog channel inputs into a common bus for analog-to-digital conversion. Analog inputs can be provided by thermocouples or strain-gain sources having a source impedance of 0 to 500 ohms. Both channel selection and gain of the multiplexer are under control of the computer; thus, the multiplexer can handle combinations of low-level and high-level analog inputs having a range from 10 mv dc full scale to 100V full scale. Channel sampling is performed at a maximum rate of 200 channels per second.

Analog inputs are converted to 12-bit digital words (11 bits plus sign) for transfer to the PDP-8/E computer. Sampling of analog inputs is initiated by the computer issuing IOT instructions. Two IOT instructions are normally required for each channel sample. The first instruction defines the multiplexer gain for the channel and loads a three-bit gain

word from the AC into a multiplexer register. The second instruction defines one of 1024 possible channels for the sample and loads an 11-bit address from the AC into a register. This instruction also starts a multiplexer timing cycle, in which system gain is established, the channel is selected, and the analog output is made available to the A/D converter. When the A/D converter completes the conversion, it loads the 12-bit word into a buffer register, terminates multiplexer sampling, and sets its device flag. The computer senses the state of the device flag by issuing ADSF instructions. When the flag is set, the A/D converter returns a skip request and the computer issues an ADRB instruction to transfer the digital word to AC00 through AC11. This instruction also clears the A/D flag to ready the device for another conversion.

The multiplexer is housed in 19-inch industrial type (H964) cabinets that have their own cooling and low-voltage power supplied. Two of these cabinets are required for the maximum channel capability of 1024 channels. The multiplexer is connected to the PDP-8/E computer using a Positive I/O Bus Interface module.

Specifications

Analog Voltage Input Ranges	10 mV full scale 50 mV full scale 100 mV full scale 200 mV full scale 500 mV full scale 1 V full scale 5 V full scale 10 V full scale 100 V full scale (with 10:1 internal attenuator)
Analog Current Ranges	1 mA full scale 5 mA full scale 10 mA full scale 50 mA full scale
A/D Converter Output Word	Parallel, 12 bits (11 bits plus sign)
Accuracy (without A/D Converter)	+ or — 25 mV or 0.5% whichever is greater.
Common Mode Rejection	120 dB or greater above 50 Hz
Common Mode Voltage	Greater than 200V
Normal Mode Rejection	50 dB or greater at 60Hz and above
System Sampling Rate	200 channels per second (max.)
Single Channel Sampling Rate	20 samples per second max. at stated accuracy
Expansion Capabilities	System can be expanded in groups of 8 channels up to a maximum of 1024 channels.

External Analog-to-Digital Converter Compatibility	Compatible with AD01 A/D Converter when converter is equipped with a post-multiplex filter.
Internal A/D Conversion Time	50 μs
Internal Converter Resolution	1 part in 2048 (4.88 mV/step)

Programming

Instructions for multiplexer operation are listed below. The ADSF and ADRB instructions are used only when the multiplexer is equipped with an A/D Converter.

Set Multiplexer Gain (ADSG)

Octal Code:	6542
Execution Time:	2.6 μs
Operation:	Loads a three-bit gain word from AC09 through 11 into a multiplexer register for selection of system gain for channel.

Set Multiplexer Address (ADSA)

Octal Code:	6544
Execution Time:	2.6 μs
Operation:	Loads an eleven-bit address from AC01 through 11 into a multiplexer register for selection of a channel. Also starts multiplexer timing to select gain and channel and provide analog sample to A/D converter.

Skip on A/D Flag (ADSF)

Octal Code:	6531
Execution Time:	2.6 μs
Operation:	Senses A/D converter flag. If flag is a one, increments the PC, and skips the next sequential instruction so the A/D converter can be serviced.

Read A/D Converter Buffer (ADRB)

Octal Code:	6534
Execution Time:	2.6 μs
Operation:	Transfers twelve-bit word in A/D buffer to AC00 through AC11, and resets A/D converter flag.

A program for selecting multiplexer gain and channel and transfer of digital word to the computer can be written as follows:

```
TAD XX
ADSG        /LOAD MULTIPLEXER GAIN WORD
CLA
TAD YY
ADSC        /LOAD MULTIPLEXER ADDRESS WORD
ADSF        /SKIP IF A/D CONVERTER FLAG IS 1
JMP .—1     /JUMP BACK AND SENSE FLAG AGAIN
```

ADRB /READ A/D BUFFER AND TRANSFER CONTENTS TO /AC00-11. CLEAR A/D CONVERTER FLAG.

Type AF04A Guarded Scanning Integrating Digital Voltmeter

The Type AF04A is a Guarded Scanning Integrating Digital Voltmeter system, with wide dynamic range and high common-mode rejection, and is capable of expansion to 1000 channels. The AF04A is used with the PDP-8/E to multiplex up to 1000 three-wire analog channels into a six-decimal digit integrating digital voltmeter (IDVM). Each digit is BCD-coded for input and display by the IDVM. Full scale ranges are from + or − 10 mV to + or − 300V, with automatic ranging, 300 percent overranging, and a usable 5 μs resolution. Guarded input construction and active integration assist in attaining an effective common-mode rejection of greater than 140 dB at all frequencies. (Normal mode rejection is infinite at multiples of power line-frequency.)

This system is ideally suited for data acquisition of process monitoring where a wide range of signals requires large dynamic range. The 10mV range has 0.001 percent resolution, and, coupled with a common-mode noise rejection greater than 140 dB at all frequencies, allows accurate direct measurement of thermocouples, strain gauges, load cells, and other low-level transducers without additional amplification.

The AF04A IDVM, operated under program control, is capable of either random channel selection or sequential channel selection. The computer selects either program-controlled ranging (for fastest speed) or auto-ranging, as well as the integration time of the integrating digital voltmeter.

The digitized data, as well as the current channel address, is read by the computer in either two or three bytes.

A decimal display of the digitized value, including sign and decimal location, is continuously displayed on the front panel. The current channel number is also displayed. Front-panel controls on the IDVM allow for manual setting of all the programmed functions. A front-panel control allows continuous display of the internal secondary standard, which can be prewired to a particular channel for reference checking during normal operation. The AF04A may be manually controlled, completely independent of the computer.

Specifications

Full scale + or −	10mV, 100mV, 1V, 10V, 100V, 300V and automatic ranging
Over-ranging	300% on all but highest range
Maximum Input Voltage	300V
Resolution	5μV (usable), 0.1μV (LSB)
Accuracy (overall worst case with daily calibration)	+ or − 0.004% of reading or + or − 0.01% of full scale
Temperature Stability (RMS full scale and zero drift)	+ or − 0.006%/day

Temperature coefficient	+ or — 0.003% of reading/degrees C
Full scale	+ or — 0.002% of full scale/degrees C
Zero	(+ or — 0.006% of full scale/degrees C on 10mV and 1V range)
Line voltage stability	+ or — 0.005%/10% change
Maximum common-mode voltage	+ or — 300V from power line ground
Common-mode rejection (166.6ms integration period and 1000-ohm source unbalance)	>140 dB at all frequencies
Normal-mode rejection	Infinite at multiples of line frequency
Input impedance	
10, 100, 1000 mV ranges	1000 megohms/V
10, 100, 300V ranges	10 megohms
Internal secondary standard	
Value	+ or — 1.000V
Accuracy	+ or — 0.002% traceable to National Bureau of Standards
Stability	+ or — 0.005%/month
Temperature coefficient	negligible

Selected Resolution

DC Voltage Range	0.001% Maximum Reading	0.001% Resolution	0.01% Maximum Reading	0.01% Resolution	0.1% Maximum Reading	0.1% Resolution
10mV	30.0000mV	0.1μV	030.00mV	1μV	0030.00mV	10μV
100mV	300.000mV	1μV	0300.00mV	10μV	00300.0mV	100μV
1000mV	3000.00mV	10μV	03000.0mV	100μV	003000.mV	1mV
10V	30.0000V	100μV	030.000V	1mV	0030.00V	10mV
100V	0300.000V	1mV	0300.00V	10mV	00300.0V	100mV
1000V*	0300.00V	10mV	00300.0V	100mV	000300.V	1V

*1000V range is scanner-limited to 300V peak maximum.

Scanning Speed
(Programmed Range)

Resolution	Integration Time	Total Time	Speed Scanning
0.1%	1.6 ms	20 ms	50 ch/s
0.01%	16.6 ms	40 ms	25 ch/s
0.001%	166.6 ms	188 ms	5 ch/s

Scanning Speed (Auto-Range)—Add 6-36 ms, depending on per-channel voltage span.

Programming

The IOT instructions associated with the scanning IDVM are designed to minimize the computer overhead associated with this option, while retaining maximum program controlled flexibility. The IOT instructions are:

Select Range and Gate (VSEL)

Octal Code: 6542
Execution Time: 2.6 μs
Operation: Transfers the contents of the accumulator to the AF04A control register. Control Word 1 is used only if a range change is required (see Figure 7-17).

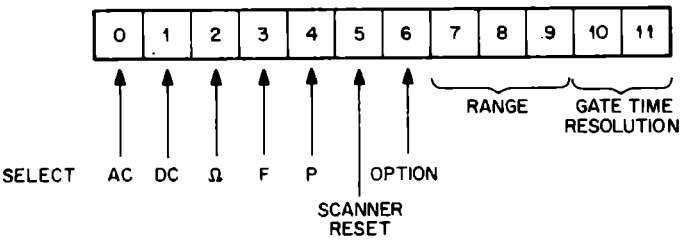


Figure 7-15 Control Word 1 (from Computer)

Select Channel and Convert (VCNV)

Octal Code: 6541
Execution Time: 2.6 μs
Operation: Transfers the contents of the accumulator to the AF04A channel address register. Automatically digitizes the analog signal on the selected channel (see Figure 7-18).

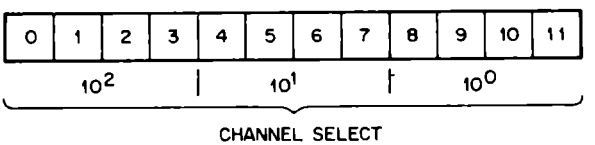


Figure 7-16 Control Word 2 (from Computer)

Index Channel and Convert (VINX)

Octal Code: 6544
Execution Time: 2.6 μs
Operation: Increments the last channel address by one and automatically digitizes the analog signal on the selected channel. The contents of the control register are unchanged.

Skip on Data Ready (VSDR)

Octal Code: 6561
Execution Time: 2.6 μs
Operation: Set a data ready flag when the scanning voltmeter has selected a channel and digitized the analog signal. This instruction is used to test for the data ready flag.

Read Data and Clear Flag (VRD)

Octal Code: 6562
Execution Time: 2.6 μs
Operation: Transfers the content of the selected byte of the IDVM output word to the accumulator and clears the data ready flag. The first data available after the flag is set is always byte 1. Subsequent bytes are program-selected using the Byte Advance command (see Figure 7-19).

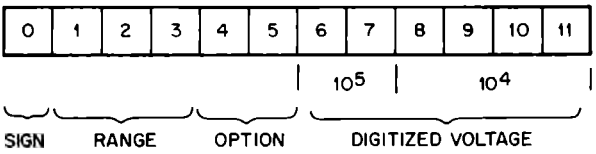


Figure 7-17 Data Word (to Computer)

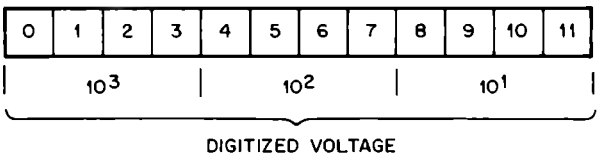


Figure 7-18 Data Word (to Computer)

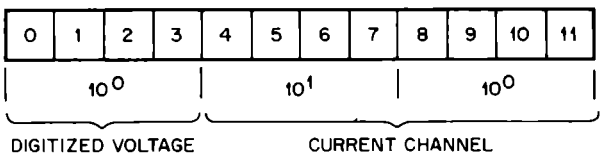


Figure 7-19 Data Word (to Computer)

All address and digitized data is in 8-4-2-1 BCD format.

Byte Advance (VBA)

Octal Code:	6564
Execution Time:	2.6 μs
Operation:	The total data word from the AF04A is 36 bits long. The first data word after the flag is set is always the 12 most significant bits. The BYTE ADVANCE command requests the next 12 most significant bits. When the data is available, the data ready flag is set again. To select the 12 least significant bits, a second BYTE ADVANCE command is required. When the data is available, the data ready flag is set again.

Sample Current Channel (VSCC)

Octal Code:	6571
Execution Time:	2.6 μs
Operation:	Digitizes the analog signal on the current channel. This command is not required except when multiple samples are required on any channel. (Using this command on a preselected channel saves up to 10 ms per sample.)

Frequency and Period Measurement Options for AF04A

A separate input permits the IDVM to be used as a frequency counter capable of counting to 2 mHz with selectable gate times of 1, 10, and 100 ms, providing measurement resolution of 10 Hz. Increased accuracy at low frequencies (to 10 kHz with automatic 250% overranging) is accomplished with the period-measurement mode. This mode counts an internal frequency source for 1, 10 or 100 periods of the frequency being measured, thereby providing increased full-scale accuracy. Period readout is in milliseconds. Frequently and voltage measurements may be made within one scanning cycle by grouping all frequency inputs in one master or slave scanner and all voltage inputs in another master or slave scanner. The output of one scanner may then be connected to the frequency-input connector of the IDVM, and the output of the other scanner to the voltage input. One of the optional control word bits is used to program the IDMV for frequency or period measurements.

Specifications (See Figure 7-20)

Frequency Measurements

Range: 10 Hz to 2 mHz

Sensitivity: 100 mV rms or −1V pulses, at least 0.3 μs wide at 50% points. 100V rms maximum working voltage.

Input Impedance: 22K shunted by less than 1000 pF, including internal cabling.

Time Base: 100 kHz crystal oscillator with initial accuracy of + or − 0.0005%, long-term stability + or − 0.001%/wk; temp. coefficient + or − 0.0002%/degrees C.

Period Measurements

Range: 1, 10, and 100 period average. Input frequency from 10 Hz to 25 kHz sine wave or 0.1 pps to 25,000 pps.

Sensitivity: 100 mV rms or −1V pulses, at least 0.3μs wide at 50% points. 100V rms maximum working voltage.
Input Impedance: 22k shunted by less than 1000 pF, including internal cabling.
Accuracy: + or − 1 count + time base accuracy + trigger error. Trigger error < + or − 0.03% for 100 mV rms sine wave with 40 dB signal-to-noise ratio.
Time Base: 100 kHz crystal oscillator with initial accuracy of + or − 0.0005%, long-term stability + or − 0.0001T/wk; temp. coefficient + or − 0.0002%/degrees C.

Selected Resolution

Selected Resolution	0.001%		0.01%		0.1%	
Function	Maximum Reading	Resolution	Maximum Reading	Resolution	Maximum Reading	Resolution
Frequency	2000.00kHz	10Hz	02000.0kHz	100Hz	002000kHz	1kHz
Period	99.9999msec	0.1μs	999.999msec	1.0μs	9999.99msec	10μs

Figure 7-20 Selected Resolution

Additional AF04A Options

A type AF04X expansion Mounting Panel is available which provides an additional 200 channels. For each 10 channels implemented, the Type AF04S 10-Channel Guarded Reed Relay Multiplexer Switch is required.

Frequency (period) measurements
AC/ohms/DC Converter
Time-of-day clock
Thumbwheel data entry panel
Thermocouple reference junctions
Extended scanner for more than 1000 channels
Special cabinet with roll-out drawer chassis accessibility.

AA50 Digital-To-Analog Converter
The AA50A DAC is a general-purpose, program-controlled DAC that converts 12-bit (11 bits plus sign bit) words into analog outputs having a full scale range of + or − 10V at 10 mA. A full-scale output range of + or − 5V at 10 mA is also available as an option.

The AA50A is housed in a H911 type mounting panel and is furnished complete with power supply, I/O cables, control and interface logic, and up to six DAC modules, each providing one analog output. The unit interfaces with the external bus of the PDP-8/E. All operations are controlled by IOT instructions, including the selection of the DAC module to receive the 12-bit output word. Each DAC module contains an output buffer register and a scaling amplifier mounted on the same module.

For an output function, the computer issues an IOT instruction that specifies the DAC module to receive the 12-bit word. The control logic

of the AA50A decodes the IOT, performs input gating for the 12-bit word from AC0-11, and loads the word into output buffer of the designated DAC module. The word remains in the output buffer until the buffer is updated by another input; thus, the resulting analog output is available until updating occurs.

Specifications

Digital Input	Parallel, 11 bits plus sign in two's complement form
Coding	3777 (octal) = + 10V 0000 (octal) = 0V 4000 (octal) = − 10V
Standard Analog Output	+ or − 10V at 10 ma
Optional Analog Output	+ or − 5V at 10 ma
Settling Time	20 μs to ½ LSB (measured at output connector with no capacitive loading)
Accuracy	.05% of full scale
Linearity	+ or − ½ LSB (+ or − 2.44mV for + or − 10V DAC output)
Capacitive Loading	0.1 μf at output connector with degradation of settling time.

Programming
The following instructions are associated with AA50A operation:

Select DAC 1 (DACS1)

Octal Code:	6551
Execution Time:	2.6 μs
Operation:	Transfers content of AC to DAC module 1 and converts it to analog output.

Select DAC 2 (DACS2)

Octal Code:	6552
Execution Time:	2.6 μs
Operation:	Transfers content of AC to DAC module 2 and converts it to analog output.

Select DAC 3 (DACS3)

Octal Code:	6553
Execution Time:	3.6 μs
Operation:	Transfers content of AC to DAC module 3 and converts it to analog output.

Select DAC 4 (DACS4)

Octal Code:	6554
Execution Time:	2.6 μs
Operation:	Transfers content of AC to DAC module 4 and converts it to analog output.

Select DAC 5 (DACS5)

Octal Code: 6555
Execution Time: 3.6 μs
Operation: Transfers content of AC to DAC module 5 and converts it to analog output.

Select DAC 6 (DACS6)

Octal Code: 6556
Execution Time: 3.6 μs
Operation: Transfers content of AC to DAC module 6 and converts it to analog output.

Device codes 56 and 57 are used when additional (up to three total) AA50's are required.

AA05A/AA07 Digital-to-Analog Converter and Control

The AA05 Digital-to-Analog Converter (DAC) provides housing power and control for up to 24 10-bit DAC modules. The AA05 Expansion Unit extends the capacity of the system to 64 channels of DAC.

Each conversion channel may use any of the four printed circuit card DAC modules. These modules include two single-buffered units, Types A608 and A609, and two double-buffered units, Types A610 and A611. A608 is a single-buffered, 10-bit DAC, with unipolar output (0V to +10V). Type A609 is a single-buffered, 10-bit DAC with bipolar output and variable offset. A610 and A611 are similar to A608 and A609, respectively, except that the former are double-buffered units.

The principal power supply furnishes all power for up to 64 DAC modules, with the exception of the −10V reference power. Reference power is furnished by the Type H706 Reference Power Supply, which is optional to the AA05/AA07 unit. A maximum of five H706 supplies can be allocated to the various DAC channels, two of which are in the AA05 and three of which are in the AA07.

The AA05/AA07 DAC and expansion unit are used with the PDP-8/E computer to control up to 64 DAC channels. Both the DAC address and the digital word to be converted are program-controlled as two I/O data words for 12-bit computers. The DAC address is stored in the AA05 and remains there until changed by the program for fast updating of any channel.

Six indicators on the front panel of this device indicate the binary address of the DAC channel currently being addressed. All data bits and I/O transfer commands are buffered to present a minimum load to the computer bus even with 64 DACs in use. The AA07 expansion assembly allows expansion to 64 single- or double-buffered DACs.

The AA05/AA07 consists of a 10-bit buffer register, level converters, a precision divider network, and a current-summing amplifier capable of driving large external loads. Provisions are made for double-buffering and bipolar output voltage where required. A precision reference voltage, supplied externally by the H706 power supply, ensures greater efficiency and optimum scale-factor matching in multiple-channel systems.

Software differences exist between the AA05/AA07 and the AA01/AA04 DAC systems. Where the AA01/AA04 employs only one basic instruction for each of three channels (DAL1, DAL2, or DAL3), the AA05/AA07 DAC utilizes four separate instructions. These instructions clear the DAC address register, transfer the contents of AC(0-9) to the input register of the selected DAC, and update all double-buffered channels (if applicable).

Specifications

Standard Output	Unipolar, 0V to + 10V at 10 ma
Optional Output	Bipolar, + or — 5V or + or — 10V
Output Impedance	Less than 1 ohm
Temperature Coefficient	0.1mV/degrees C plus temperature coefficient of reference supply (worst-case for DEC reference supply is 0.6mV/degrees C)
Resolution	0.1% of full-scale
Accuracy	+ or — 5mV
Settling Time (Full-scale)	5 μs for 1 DAC module. Less than 100 μs for up to 12 DAC modules
Environmental	0 degrees to 50 degrees C
Power	7A (max) at 115V, 60Hz

Programming
The following instructions are associated with the AA05A DAC:

Clear DAC Address (DACL)

Octal Code:	6551
Execution Time:	2.6 μs
Operation:	Clears DAC address register.

Load DAC Address (DALD)

Octal Code:	6552
Execution Time:	2.6 μs
Operation:	Loads content of AC in DAC address register.

Load DAC Input Register (DALI)

Octal Code:	6562
Execution Time:	2.6 μs
Operation:	Loads content of AC in DAC input register specified by DAC address register.

Update All Channels (DAUP)

Octal Code:	6564
Execution Time:	2.6 μs
Operation:	Updates all double-buffered channels to provide DAC outputs to loads.

Universal Digital Controller (UDC)
The UDC is a digital input/output system with a controller having 256 12-bit addressable channels. Each channel can be used as an input or output path. When used for output functions, a channel can control 12 discrete off/on devices such as relays, flip-flops, etc. When used for input functions, a channel can be used to interrogate the status of

12 discrete off/on sources such as switches, relays, and flip-flops. Thus, the UDC provides the capability for accessing a total of 3072 (10) discrete digital points either for input (status) or output (control) functions in 12-bit combinations.

All input/output data is handled in the form of 12-bit words. The data is unstructured except for the generic module type and address word read to the computer after an interrupt. Accumulator bits 0 through 3 receive a four-bit code denoting the generic type or function performed by the eight-bit address specified in AC04 through AC11.

Any UDC channel or word can be input or output. When dedicated for an input function, the type of interrupt desired must be specified by the program. The type of interrupt is defined by AC10 and AC11 as follows:

AC10	AC11	TYPE OF INTERRUPT
0	0	None
0	1	Deferred processing
1	0	Immediate processing
1	1	Both

Once an interrupt type is selected and an interrupt occurs, the UDC locates the interrupting address, using an address scan cycle. This cycle requires approximately 50 μs. Once the interrupting address has been located, the address and module generic type are made available to the computer.

The functional capability to EXCLUSIVE OR an AC word with an I/O word is provided by Change-Of-State (COS) gating. The AC bits are loaded into the COS register with the Load Previous Status (octal 6357) IOT. Data from the word of I/O presently addressed is at the gates and the EXCLUSIVE-OR function is performed.

Two bits, pulse open and pulse close, are hard-wired at the I/O word in question; their purpose is to mask out data changes that are not pertinent. The EXCLUSIVE-OR function is defined by the following:

DATA BIT	AC BIT	PULSE OPEN	PULSE CLOSE	COS OUT
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

The UDC is housed in an H964 industrial-type cabinet(s) complete with cooling fans and low-voltage power supplies. The smallest configuration of the UDC consists of one file in a single cabinet. This file contains the IOT and interface control logic; the address scanner logic, and capabilities for handling up to four I/O channels or words. This basic system can be expanded from four channels to 256 channels in four-channel groups by adding system units, file and cabinets.

Specifications

Operating Modes	Digital Output Digital Input Interrupt or Noninterrupt Change of State/Direction
Data Format	Parallel, 12-bit unstructured
Addressing Capability	256 12-bit channel or words or 3072 discrete digital points
Input/Output Module Selection	Directly addressable and location independent
Interrupt Module Identification	4-bit Generic code type
Interrupt Structure	Immediate or deferred by module assignment and program
Interrupt Scan or Address Location Time	25 μs (max)
I/O Data Rate	1,000,000 12-bit word per second
System Clock Rates	3 clock rates available to each I/O channel or word (1) 60 Hz, 6.3 VAC (line power) (2) 175 Hz to 1.75 kHz (3) 1.75 Hz to 17.5 kHz
Standard Output Drive Capabilities	250 ma at 55V (suitable for relay drivers)
Optional Output Drive Capabilities	2 amps, 250V, 100VA (for Mercury-wetted relays)
Standard Inputs	15 ma. at 6V
Optional Input	15 ma. at 24V or 48V

Programming

The following instructions are associated with UDC operation:

Skip On Scan Not Busy (UDSS)

Octal Code:	6351
Execution Time:	2.6 μs
Operation:	Skips the next instruction if Scan Not Busy flag is a one, denoting that the address scanner has located the interrupt channel, so that UDC can be serviced.

Start Interrupt Scan (UDSC)

Octal Code:	6353
Execution Time:	3.6 μs
Operation:	Enables address scan function if interrupt flag is set and interrupt type (immediate or deferred) is present.

Read Address and Generic Type (UDRA)

Octal Code: 6356
Execution Time: 3.6 μs
Operation: Transfers the generic type and address to the AC after interrupting address has been located.

Load Previous Status (UDLS)

Octal Code: 6357
Execution Time: 4.6 μs
Operation: Loads content of AC into COS register and reads the result of the EXCLUSIVE OR function of the COS logic to AC.

Skip On UDC Flag and Clear Flag (UDSF)

Octal Code: 6361
Execution Time: 2.6 μs
Operation: Skips the next instruction and clears the UDC flag if UDC interrupt Flag is set.

Load Address (UDLA)

Octal Code: 6363
Execution Time: 3.6 μs
Operation: Loads 8-bit address from AC into address register scanner.

Enable UDC Interrupt Flag (UDEI)

Octal Code: 6364
Execution Time: 2.6 μs
Operation: Sets the interrupt enable flip-flop so that UDC can generate interrupt requests.

Disable UDC Interrupt Flag (UDDI)

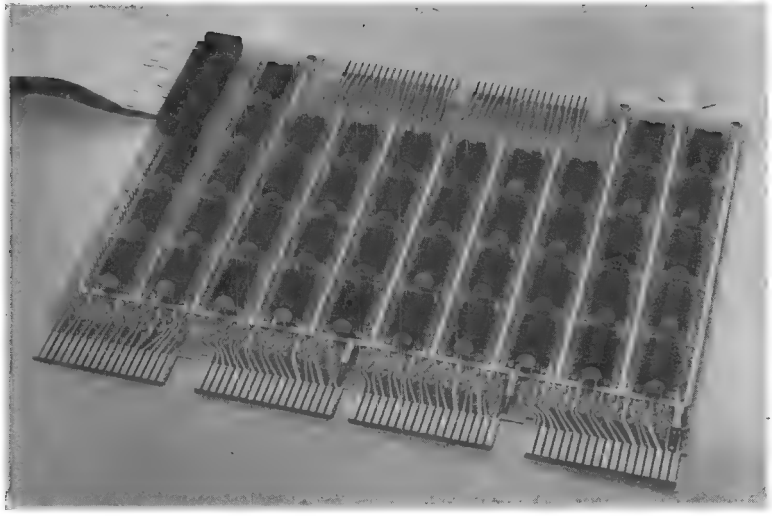
Octal Code: 6365
Execution Time: 3.6 μs
Operation: Clears interrupt enable flip-flop so that UDC cannot generate interrupt requests.

Clear AC and Read Data (UDRD)

Octal Code: 6366
Execution Time: 3.6 μs
Operation: Clears the AC and transfers data specified by address of address scanner register to AC.

Load Data and Clear AC (UDLD)

Octal Code: 6367
Execution Time: 4.6 μs
Operation: Transfers content of AC to address specified by address scanner register, then clears AC.



Type W967 OMNIBUS Interface Module—for user desiring to design and construct limited quantity or prototype interface circuits.

part 3

INTERFACING AND INSTALLATION

CHAPTER 8
DIGITAL LOGIC CIRCUITS

INTRODUCTION

The digital logic circuits in this chapter are used to interface I/O devices to the computer using Digital Equipment Corporation FLIP CHIP Modules. Logic handbooks published by DEC describe hundreds of FLIP CHIP Modules with their component circuits, associated accessories, hardware, power supplies, and mounting panels. The designer should study the logic handbooks carefully before beginning on interface design for a special I/O device.

The basic logic circuits used for interfacing to the computer are: AND, OR, NAND, NOR, Flip-Flop, Single-Shot, Schmitt Trigger, Inverter, Amplifier, and Bus Driver. A brief discussion of these circuits and their logic symbology follows.

The symbology employed with the PDP-8 family of computers and M-series modules is similar to MIL-STD-806B. This chapter describes DEC symbology with definitions of logic functions, graphic representations of the functions, and examples of their application. A Table of Combinations is also shown.

LOGIC SYMBOLS

The following description of logic symbols contains truth tables that show graph representations of the logic functions. In the truth tables, the letter H stands for HIGH (+ 3V), and the letter L stands for LOW (0V). Examples of DEC symbology are shown along with figures and truth tables.

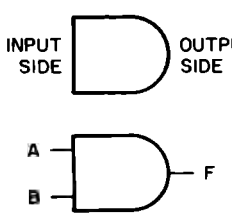
State Indicator

The presence of the small circular symbol at the input(s) of a function indicates that an L input signal activates the function. The absence of this small circle indicates that an H input signal activates the function. Similarly, a small circle at the output of a function indicates that the output terminal of the activated function is relatively low, and the absence of the circle indicates that the output is relatively high.

STATE INDICATOR ABSENT

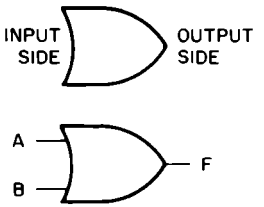
a. AND—The symbol in Figure 8-1 represents the AND function. The output (F) is high only if both inputs (A and B) are high.

b. OR—The symbol in Figure 8-2 represents the OR function. The OR output (F) is high if any input (A or B) is high.



INPUT		OUTPUT
A	B	F
L	L	L
L	H	L
H	L	L
H	H	H

Figure 8-1 Symbol, AND Function

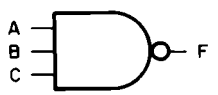


INPUT		OUTPUT
A	B	F
L	L	L
L	H	H
H	L	H
H	H	H

Figure 8-2 Symbol, OR Function

STATE INDICATOR PRESENT

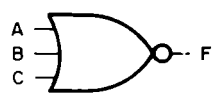
a. NAND—The symbol in Figure 8-3 represents the NAND function. The output (F) is low only when all inputs (A, B, and C) are high. NAND logic is the major gate configuration in the PDP-8 family.



INPUT			OUTPUT
A	B	C	F
L	L	L	H
L	L	H	H
L	H	L	H
L	H	H	H
H	L	L	H
H	L	H	H
H	H	L	H
H	H	H	L

Figure 8-3 Symbol, NAND Function

b. NOR—The symbol in Figure 8-4 represents one version of the NOR function. The output (F) is low if one or more of the inputs (A, B, and C) are high.

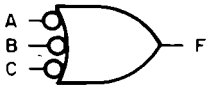


INPUT			OUTPUT
A	B	C	F
L	L	L	H
L	L	H	L
L	H	L	L
L	H	H	L
H	L	L	L
H	L	H	L
H	H	L	L
H	H	H	L

Figure 8-4 Symbol, NOR Function

c. NOR—The symbol in Figure 8-5 represents another version of the NOR function. The output (F) is high if one or more of the inputs

(A, B, or C) are low. The NOR version for this function is identical to one version of the NAND function shown in Figure 8-3.



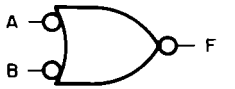
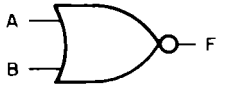
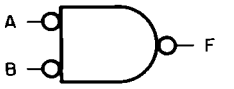
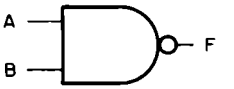
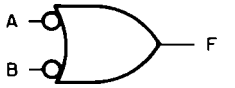
INPUT			OUTPUT
A	B	C	F
L	L	L	H
L	L	H	H
L	H	L	H
L	H	H	H
H	L	L	H
H	L	H	H
H	H	L	H
H	H	H	L

Figure 8-5 Symbol, NOR Function

Table of Combinations

Table 8-1 illustrates the applications, functions, and truth tables of two variables and their equivalents, as well as their relationship to DEC logic.

Table 8-1 Table of Combinations

AND	OR	A	B	F
		H	H	H
		H	L	L
		L	H	L
		L	L	L
		H	H	L
		H	L	L
		L	H	L
		L	L	H
		H	H	H
		H	L	H
		L	H	H
		L	L	L
		H	H	L
		H	L	H
		L	H	H
		L	L	H

The symbol in Figure 8-6 shows the flip-flop function. The pins are numbered counterclockwise on a standard flip-flop. The flip-flop has four possible inputs: set (S), reset(R), data(D), and clock (C); and two data outputs, logic 0 (low) and logic 1 (high). If the data input is high and a pulse is applied to the clock input, the flip-flop sets to the logic 1 state. If the data input is low and a pulse is applied to the clock input, the

flip-flop resets to the logic 0 state. When the data input is shown with a small circle (redefined flip-flop), the opposite of the above is true; that is, if the data input is high and a pulse is applied to the clock input, the flip-flop goes to its logic 0 or reset state, etc.

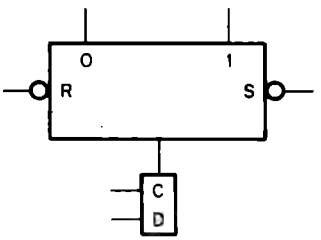


Figure 8-6 Symbol, Flip-Flop Function

One-Shot Function

The symbol in Figure 8-7 shows the one-shot (OS) function. The output signal shape, amplitude, duration, and polarity are determined by the circuit characteristics of the OS device. When it is not activated, the one-shot device is in either a 0 or 1 state. When the input is pulsed by a high-to-low level change, the 1 output goes high and remains high; and the 0 output goes low and remains low for the specific time of the device.

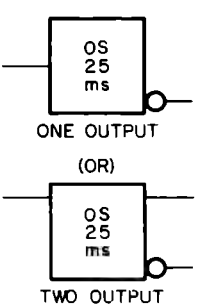


Figure 8-7 Symbol, One-Shot Function

Schmitt Trigger

The symbol in Figure 8-8 shows the Schmitt trigger (ST) function. The Schmitt trigger is normally either in a 0 or a 1 state (inactivated). When the input signal crosses a predetermined voltage threshold, the Schmitt

trigger changes to its opposite state and remains there until the input signal falls below the threshold.

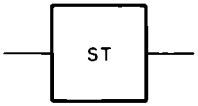


Figure 8-8 Symbol, Schmitt Trigger Function

General Logic Symbols

The symbol in Figure 8-9 is used for functions not specified elsewhere. An example of this symbology is the inhibit driver used in the PDP-8 family. When possible, an explanation or abbriviation of the function is contained within the box.

Figure 8-9 Symbol, General Logic

Amplifier

The symbol in Figure 8-10 shows a linear or nonlinear current or voltage amplifier. This symbol is used to represent level changers, inverters, emitter-followers, and lamp drivers.

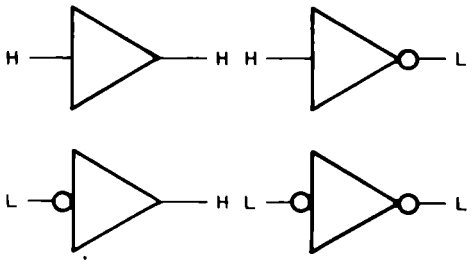


Figure 8-10 Symbol, Amplifier

Time Delay

The symbol in Figure 8-11 shows a time-delay device. The time-delay duration is specified inside the symbol unless there are two or more outputs. When there are two or more outputs, the delay time of each output is marked adjacent to that output.

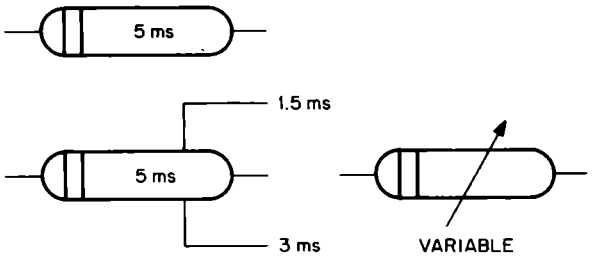


Figure 8-11 Symbol, Time Delay Function

CHAPTER 9

THE OMNIBUS INTERFACING

INTRODUCTION

Interfacing to the PDP-8/E OMNIBUS is accomplished with both hardware and software. For standard interfaces, DEC supplies necessary option modules and the equivalent programs (subroutines and MAINDECS) necessary to perform those DEC defined functions. This software is quite adequate to satisfy the operational requirements of each standard option as defined in Chapter 7. However, in the event that the user desires program functions different from those given in the software packages, he must change the software using the standard input/output devices such as a teleprinter, a card reader, or the console switches. Before undertaking this, however, the user should acquaint himself with the software routines (good documentation of the software is provided with each program tape) and read INTRODUCTION TO PROGRAMMING, a volume in the DIGITAL small computer handbook series.

This chapter provides the necessary information for users desiring to build a special interface. It deals primarily with the hardware consideration and it is understood that the user must create his own program for his own defined functions. If the user lacks sufficient experience to design his interface, he can contact his local DEC Sales Office for special assistance.

The means of transferring commands and signals from module to module is accomplished on what is called the "OMNIBUS." All PDP-8/E modules, including options, plug into the OMNIBUS in a significantly accessible manner.

The OMNIBUS is an etched board with rows of connectors soldered to the board. The pin assignment is the same on all connectors. Thus, the OMNIBUS accommodates 96 signals, which feed to 96 pins on the connectors. The user is generally only concerned with those signals that control data transfers, address memory, or contain the data to be transferred. However, the additional signals, such as timing, are readily available on the OMNIBUS to accommodate any tailor-made requirement in the event that the user should design and build his own interface module.

Many advantages are derived from the OMNIBUS approach. Because all connectors on the OMNIBUS contain the same signals, a module can be placed anywhere on the bus at the convenience of the user. All random wiring is eliminated with this type of arrangement. This feature provides greater performance, and reliability. Considerable space is conserved; thus providing a unique packaging capability that allows a high density of electronic circuitry in a small area.

The information in chapter 9 is presented in two sections. The first section describes the many aspects of the OMNIBUS in terms of its

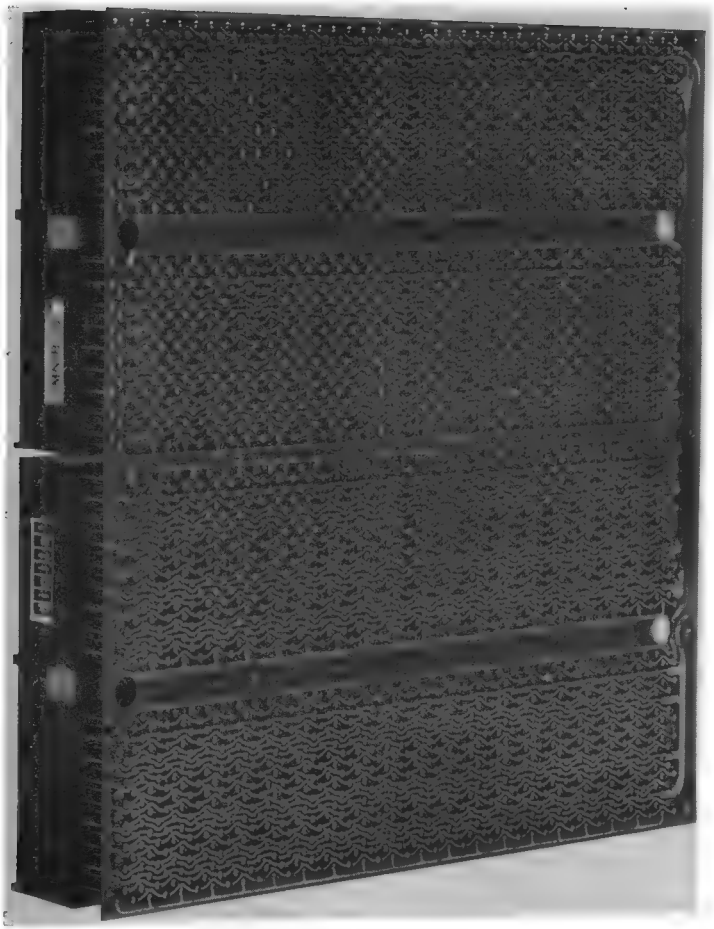


Figure 9-1 PDP-8/E OMNIBUS

physical qualities, the types of interfaces placed on the bus, and the input/output transfer schemes. This information is intended to provide the reader with sufficient background to begin designing an interface control module.

Section 2 is devoted strictly to the designing and constructing of an interface control module. Included are rules, guide lines, and suggestions from DEC engineers who designed the PDP-8 family of computers.

SECTION 1 OMNIBUS DESCRIPTION

BUS STRUCTURE

The OMNIBUS (H919 OMNIBUS Assembly) is a back plane etched circuit board with ten H803 connectors mounted onto the board and wave soldered. The circuit board is $10\frac{1}{2}$ inches by $10\frac{1}{2}$ inches with a $1\frac{5}{8}$ inch thickness. The OMNIBUS is attached to the bottom of the PDP-8/E mounting box and is the means by which all modules are connected. Figure 9-1 shows the OMNIBUS with all connectors mounted. A single assembly accommodates 20 PDP-8/E modules.

The OMNIBUS is designed so that all back plane wiring is eliminated and so that every pin in a given connector slot is defined. All modules plugging into the bus are PDP-8/E modules. If a functional unit on the bus requires more than one module, Type H851 edge type connectors on the top of the board connect multiple boards together. For cables to the "outside world," connectors on the side of the module connect to a shielded coaxial or flat ribbon cable. In this arrangement, up to 2 connectors for each module may be used.

Figure 9-2 shows the OMNIBUS with modules plugged into it. Each module functional unit can be placed anywhere on the bus or removed from the bus without affecting the operation or performance of the system.

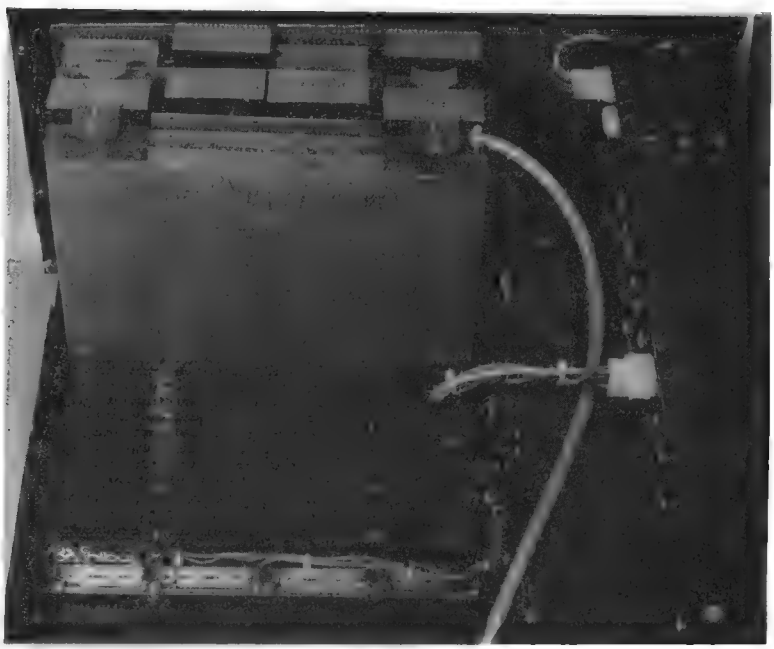


Figure 9-2 PDP-8/E Modules mounted on the OMNIBUS.

Each OMNIBUS contains 20 slots; two assemblies with a M935 bus connector provides 40 slots, two of which are used to interconnect the assemblies. Thus, 38 slots are available when the OMNIBUS Expansion unit is used. However, the OMNIBUS can be expanded to accommodate an additional 38 modules. This is illustrated in Figure 9-3, which shows the basic OMNIBUS connected to the OMNIBUS expansion unit.

Figure 9-3 The System Expansion Capability.

BUS SPECIFICATIONS
Logic Levels

Logic 1—Max Voltage: 0.4V
Min Voltage: -0.5V

Logic 0—Max Voltage: 5.0V
Min Voltage: 3.0V

SYSTEMS CONFIGURATION
The PDP-8/E with all primary options is identified in Figure 9-4. The basic system contains the central processor (4 modules) the programmer's console (1 module), 4K memory (3 modules) and a console Teletype control (1 module).

System expansion is easily accomplished simply by adding "off-the-shelf" control units necessary to accommodate the corresponding peripheral equipment, if additional machine capability is desired. For example, if it were desired to add additional memory capability to a basic PDP-8/E, a Memory Extension Control and Time Share Option, type KM8-E, could be added. Then 4K memory units could be added, up to a maximum 32K capability. For those customers who wish to use PDP-8/I or PDP-8/L compatible peripherals, an external bus option such as the Positive I/O Bus Interface Module, type KA8-E, and the Data Break Interface Module, type KD8-E, connects to the OMNIBUS to provide interfacing capabilities.

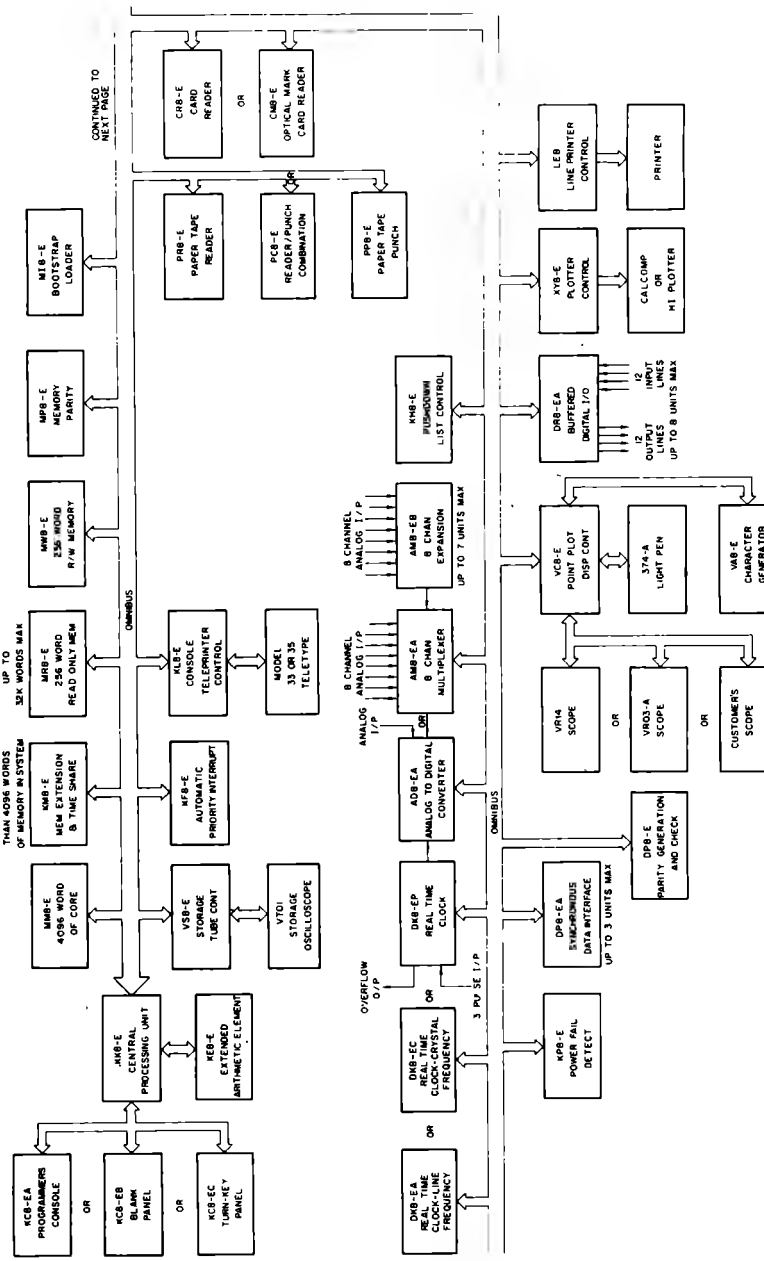


Figure 9-4 PDP-8/E—OMNIBUS Configurator

Almost all types of peripherals are included as an option to allow the user to expand his system to his own requirements. However, in the event that the user has a unique requirement such as a special control system, he may build his own control module by following the rules specified in this chapter. Refer to Chapter 11 for planning and installation.

RELATIONSHIP OF THE EXTERNAL BUS TO THE OMNIBUS

The External Bus, which is mechanically and electrically organized the same as the I/O bus on the PDP-8/L or the PDP-8/I with KA8-I, plugs into the OMNIBUS by way of the Positive I/O Bus interface and the Data Break interface. Each of these modules receives the same signals on the same pins as any other module plugged into the OMNIBUS. The interfacing details to the External Bus are given in Chapter 10 of this handbook.

OMNIBUS SIGNALS

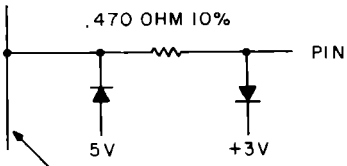
The signals and pin assignments of the OMNIBUS are given in Figure 9-5. Bus loads are provided in Figure 9-6. Each load corresponds to a description of each signal that is provided in Table 9-1. The Loading Rules presented later in this chapter provide information on the electrical properties of these lines.

PIN	A1	A2	B1	B2	C1	C2	D1	D2
A	TP	+5V	TP	+5V	TP	+5V	TP	+15V
B	TP	-15V	TP	-15V	TP	-15V	TP	-15V
C	SP GND*	GND	GND	GND	GND	GND	GND	GND
D	MA0	EMA0	MA4	INT STROBE	I/O PAUSE	TP1	MA8	IR0
E	MA1	EMA1	MA5	BRK IN PROG	C0	TP2	MA9	IR1
F	GND	GND	GND	GND	GND	GND	GND	GND
H	MA2	EMA2	MA6	MA, MS LOAD CONT	C1	TP3	MA10	IR2
J	MA3	MEM START	MA7	OVERFLOW	C2	TP4	MA11	F
K	MD0	MD DIR	MD4	BREAK DATA CONT	BUS STROBE	TS1	MD8	D
L	MD1	SOURCE	MD5	BREAK CYCLE	INTERNAL I/O	TS2	MD9	E
M	MD2	STROBE	MD6	LD ADD ENABLE	NOT LAST XFER	TS3	MD10	USER MODE
N	GND	GND	GND	GND	GND	GND	GND	GND
P	MD3	INHIBIT	MD7	INT IN PROG	INT RQST	TS4	MD11	F SET
R	DATA0	RETURN	DATA 4	RES 1	INITIALIZE	LINK DATA	DATA8	LOAD ADDRESS
S	DATA1	MEMORY READ	DATA 5	RES2	SKIP	LINK LOAD	DATA 9	STOP
T	GND	GND	GND	GND	GND	GND	GND	GND
U	DATA 2	ROM ADDRESS	DATA 6	RUN	CPMA DISABLE	IND 1	DATA 10	KEY CONTROL
V	DATA 3	LINK	DATA7	POWER OK	MS, IR DISABLE	IND 2	DATA 11	SW

* THIS PIN IS CONNECTED TO GROUND ON THE BUS, BUT SERVES AS A LOGIC SIGNAL WITHIN MODULES TO FACILITATE TESTING

Figure 9-5 OMNIBUS Pin Assignment

Figure 9-6a LOAD 1



COMMON LINE
WHICH IS +15V
DURING TP4
OPEN-CIRCUITED OTHERWISE

Figure 9-6b LOAD 2

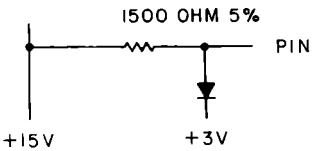
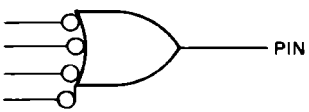


Figure 9-6c LOAD 3



TTL GATE, USUALLY SN7440
DO NOT GROUND WITH OPEN COLLECTOR

Figure 9-6d LOAD 4

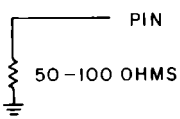


Figure 9-6e LOAD 5

Figure 9-6 OMNIBUS Loads

Table 9-1 OMNIBUS Signals

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD
Parallel Data & Addressing	MA0-11	Used to address memory to any one of 4096 possible locations. This address is changed only at TP4 Time. The address is normally developed in the processor. However, during a data break, the MA lines can be used for break addresses, which originate in the data break module. When the processor is executing an instruction, the address always originates in the processor.	2
	EMA0-2	Used only when the extended memory is provided. These 3 bits are combined with the 12 bit Memory address to form a 15 bit memory address. This is necessary to specify one location out of 32,768 possible locations. The extended address bits specify the memory field in use. MA11 is the least significant bit and EMA0 is the most significant bit. All 12 or 15 lines are high for a zero and low for a one. Thus, if the machine does not contain a Memory Extension Control, the EMA bits are automatically zero (high), selecting the lowest 4K of memory.	2
	MD0-11	The 12 Memory Data lines carry information to and from memory. The direction of data flow on these lines is controlled by a signal called "MD DIR." If MD DIR is held low, information stored in the sense flip-flops of the currently active memory is placed on the MD lines. When MD DIR is high, the contents of the MD register in the CPU are placed on the MD lines. Information that was in memory or is to be placed into memory is on these memory data lines. Types of information carried on these lines include operands, information being stored in memory, information that is read from	3

Table 9-1 OMNIBUS Signals (cont)

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD
		memory, select codes, etc. Information that is transferred out of memory to a peripheral device during a data break would be transferred during the first half of the memory cycle. Information that is transferred from a peripheral device into memory during a data break would be placed into memory during the second half of the memory cycle. As in the case of the MA lines, the MD lines are low for a one and high for a zero. The MD lines provide the peripheral with its device code through the middle six bits (bits 3 thru 8). The MD lines are the only connecting data link between the processor and memory.	
	DATA 0-11	DATA bits 0-11 serve as a bidirectional multipurpose bus. Generally, these lines are the in/out path between the peripheral and the AC register. The primary signals that allow data transfer on these lines are called C0, C1, and C2. During TS1, the DATA lines carry information for the front panel (see IND1 and IND2). During TS2 of the break cycle, they carry information to be loaded into the MB. During TS3 of Operate and time states 2 and 3 of IOT instructions, they carry the contents of the AC. During TS4 they carry Break or API priority. As in the case of the MD and MA lines, DATA lines are low for a one and high for a zero.	3
Memory Control Signals	MEM START	This line is grounded for a minimum of 100 ns to initiate a memory cycle. It must not be grounded after TP2. Memory cycles continue automatically until STOP is grounded.	3
	MD DIR	This signal defines the data direction on the MD lines. If this signal	3

Table 9-1 OMNIBUS Signals (cont)

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD
		is low, the data on the MD lines originates in the sense register of the selected memory. If this signal is high, the MD lines are driven from the MB. MD DIR does not necessarily change in the middle of each memory cycle. If, for example, a Fetch cycle is being executed, MD DIR is low for the entire cycle. Thus, the contents of memory are rewritten from the local sense register. The, state of MD DIR should be changed only at TP1 or TP2.	
	ROM ADDRESS	When this line is high, the Read/Write memory runs normally. When this line is low, the Read/Write memory does not function, despite memory timing signals on the bus. This line is used when a small ROM is used with addresses that overlap the main memory. If the ROM sees its address on the MA lines, this line is immediately grounded. If ROM ADDRESS is grounded during the Execute portion of a JMS instruction, the next instruction will be taken from the addressed location, thus saving a ROM location.	2
	MEMORY WRITE, SOURCE, STROBE, IN-HIBIT, RETURN	These five signals control the memory, and may vary if different memory and timing modules are used. MEMORY WRITE is high during the write portion of the memory cycle. SOURCE is used to turn on memory current. It is high when read or write current is to be turned on. RETURN and SOURCE turn on (go high) at the same time, but RETURN turns off 50 ns later to insure that the stack does not remain capacitively charged. INHIBIT turns on the inhibit drivers when positive. STROBE provides a time reference from which the output of the sense amplifiers are	4

Table 9-1 OMNIBUS Signals (cont)

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD
		sampled. STROBE goes positive before data is actually ready in the sense amplifiers. Each memory then delays this leading edge by the optimum amount. This precaution allows for stack variation. If data is read from memory, the negative going edge of STROBE indicates the data on the MD lines is valid.	
Programmed I/O Signals	I/O PAUSE	I/O PAUSE is asserted (brought to ground) at TP2 of an IOT cycle. This signal is driven by a bus driver since the load is heavy. The I/O PAUSE line signifies that an IOT is being processed. It is used by the peripheral control modules to gate device selectors and to gate data onto the MD lines, and is ANDed with TP3 to initiate output transfers. The signal is negated at the last I/O transfer. (See Last Transfer.)	4
	C0, C1, C2	These lines control the type of I/O transfer in accordance with the following table:	3
		C0 C1 C2 Function	3
		High High High DATA V AC goes to AC	
		Low High High AC goes to DATA, 0 goes to AC	
		High Low High DATA V AC goes to AC	
		Low Low High DATA goes to AC	
		* High Low DATA + PC goes to PC	
		Low Low DATA goes to PC	
		* C0 is ignored by the CP whenever C2 is low.	
	BUS STROBE	BUS STROBE is a negative-going pulse that signifies an I/O transfer is taking place. It is this pulse, together with the proper state(s) of C0, C1, C2, that causes data exchanges. Unless NOT LAST XFER	3

Table 9-1 OMNIBUS Signals (cont)

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD
		is ground, TP3 automatically generates BUS STROBE and terminates the transfer. The I/O structure of the bus is such that several BUS STROBES may take place in a single IOT, if desired. To initiate such a transfer, NOT LAST XFER must be ground at TP3 time. The final BUS STROBE, made with NOT LAST XFER high, terminates the I/O exchange. C0, C1 and C2 must be set up a minimum of 280 ns before BUS STROBE. For C1 high and C2 low, this set-up time is 400 ns.	
	INTERNAL I/O	Internal peripherals normally exchange information at memory speeds; whereas peripherals on the external, PDP-8 family compatible I/O bus must operate more slowly to allow adequate IOP propagation and signal return times. The INTERNAL I/O line is brought to ground by a peripheral when it sees I/O PAUSE and its device code(s). If INTERNAL I/O is ground, the IOT is ignored by the positive I/O Bus interface module.	3
	NOT LAST XFER	A ground level on this line indicates to the CP that the next BUS STROBE does not terminate the I/O transaction. Since most internal I/O devices use only one transaction per IOT, this signal is normally not grounded by the internal I/O devices. Thus, the internal peripheral usually only asserts its "C" lines and INTERNAL I/O.	3
	SKIP	This signal is sampled by the CP at TP3 of an IOT or operate instruction. It causes PC + 1 to go to MA at TP4 time.	3
Interrupt Signals	INT RQST	INT RQST is used by all peripherals to cause an interrupt. This line is normally high, and is brought to	

Table 9-1 OMNIBUS Signals (cont)

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD
		ground by the requesting device. It is sampled at the CP at Interrupt Strobe time, and causes the CP to do a JMS to location 0 of memory 0 as its next instruction if the interrupt system is enabled.	
	INT IN PROG	INT IN PROG signifies that the CP is in the process of honoring an interrupt request. This line is asserted (brought to + 3V) at INT STROBE time if all conditions for granting an interrupt request are present. It is negated at TP1 of the next Execute and JMS cycle. Grounding this line prevents honoring an interrupt, even though all other conditions are met.	3
PROCESSOR STATES AND SIGNALS	IR0, 1, & 2	These 3 lines indicate the effective instruction being processed. They usually, (but not always), indicate the contents of the IR. Lines are low for a 1 and high for a 0. All instruction processing and major state gating is derived from these lines.	3
	F, D, E	The 3 major states lines, like the IR lines, indicate the effective major state. The appropriate line is low to indicate its major state. Only one of these lines should be grounded at any time. No major state is a legitimate state of the machine—this state is used by the front panel and by break devices.	2
	F SET	F SET is similar to major states described above. This line must not be grounded by modules external to the CP. F SET indicates the next machine cycle is a fetch unless disabled. Note that No Major State causes F SET.	4
	LINK LOAD, LINK	These two lines may be used to jam one bit of information into the LINK. LINK DATA is low for a	LINK LOAD: 1

Table 9-1 OMNIBUS Signals (cont)

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD
	DATA	1 and high for a zero. LINK LOAD is normally high and is brought to ground to cause loading.	LINK DATA: 3
	LINK	This line indicates the state of the link bit in the processor. It is high if the link is 0, and low if the link is a 1.	3
Panel Control Signals	LD ADD ENABLE	When this signal is asserted (grounded), a date path is enabled to allow the DATA bus to be loaded into the MA.	3
	KEY CONTROL	This line, when asserted (grounded), causes CPMA + 1 to go to PC and PC to go to CPMA if a memory cycle occurs. Also, if LOAD ADDRESS is asserted, the contents of the DATA bus are loaded into the Extended Memory Control if KEY CONTROL is low, or into the CPMA if KEY CONTROL is high. If asserted, the RUN flip-flop is cleared at TP3. Key Control prevents interrupts from occurring when low.	3
	STOP	STOP is asserted (grounded) by the STOP key and F SET, by the SINGLE STEP key, by KEY CONTROL (low), or by the HLT instruction. It is sampled at TP3. If this line is low, and processing is stopped at the end of the current memory cycle.	3
	LOAD ADDRESS	This is a positive pulse that either causes DATA to go to MA (if KEY CONTROL is high), or causes DATA 6 thru 11 to go to IB, IF, DF (if Key Control is low). Note that LOAD ADDRESS does not initiate a memory cycle.	5
	IND 1 & IND 2	These 2 lines control the data placed on the DATA lines at TS1 time.	3

Table 9-1 OMNIBUS Signals (cont)

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD

IND 1 / IND 2 / Effect

IND 1	IND 2	Effect
High	High	Status word goes to DATA lines at TS1
High	Low	C (MQ) goes to Data lines at TS1
Low	High	DATA lines to lights. (No time state)
Low	Low	C (AC) goes to Data lines at TS1

Status word format:

Bit	Function	Front Panel Appr.
0	Link	L
2	Interrupt Bus	GT
1	"Greater than" F/F	INT BUS
3	No Int. Allowed	NO INT.
4	Interrupt On	ION
5	User Mode	UM
6	Instruction Field 0	IF 0
7	Instruction Field 1	IF 1
8	Instruction Field 2	IF 2
9	Data Field 0	DF 0
10	Data Field 1	DF 1
11	Data Field 2	DF 2

	SW	SW is a line controlled by front panel switch SW. When the switch lever is up, the line is high. When the lever is down, the line is low.	3
Data Break Signals	BREAK IN PROG-RESS	This line is asserted (grounded) at INT STROBE time if a break request is recognized. All break devices making a request place their priorities on the DATA bus at TS4. This line also causes the BREAK IN PROGRESS light to be lit on the front panel.	3
	MS, IR DISABLE	The state of this line is changed at TP4. Asserting (grounding) this line causes the MAJOR STATE and the IR outputs to be disconnected from their lines on the bus and	2

Table 9-1 OMNIBUS Signals (cont)

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD
		hence from all circuitry within the CP. F SET is asserted, allowing the next loading of the major state register to place the CP in the Fetch cycle. This feature is used by the front panel, but not by Data Break devices since Data Break devices ground MA, MS LOAD CONTROL. MS, IR DISABLE is used by the front panel to force the CP to ignore its former state. It is also used by break devices to inhibit processing while a break is in process. In addition, MS, IR DISABLE also enables DATA (+ MD) to MB.	
	CPMA DISABLE	Asserting (grounding) this line at TP4 causes the CP's Memory Address register to be disconnected from the MA lines. Simultaneously in a Break cycle, the BKMA register of the highest priority device is gated onto the MA bus. CPMA DISABLE also modifies the E cycle of a JMS instruction so that the next instruction is taken from the addressed location. (See KH8-E option).	2
	MA, MS LOAD CONTROL	MA, MS LOAD CONTROL is changed at TP1. When this line is grounded, the CPMA and major state registers are not loaded at TP4.	3
	OVER-FLOW	This line is driven by a flip-flop that senses the carry from the adders at TP2. The flip-flop is set each time there is a carry or borrow out of the MB, and is "ANDed" with TS3 before going to the bus. Although this flip-flop is set by carry from an ISZ instruction, its primary use is in WC and B cycles. Data Break interface modules gates OVERFLOW with its major state to determine Word Count Overflow, overflow with ADM, and borrow with ADM.	3

Table 9-1 OMNIBUS Signals (cont)

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD
	BREAK DATA CONT	Controls the loading of the MB when MS and IR are disabled. If this line is high, DATA goes to MB. If this line is low, DATA + MD goes to MB. This line must not be changed during TS2 and is usually changed at TP4.	3
	BREAK CYCLE	BREAK CYCLE is used by break devices to drive a front panel indicator. When ground, the true B cycle is in progress.	3
Timing Signals	TP1, TP2, TP3, TP4	These 100 ns positive-going pulses originate in the timing module. The exact spacing of the timing pulses is a function of fast or slow cycle. The source of all timing is a 20 MHz crystal clock and a frequency divider. The timing generator starts running any time Mem Start is issued, and continues to run until TP4 occurs. At that time, if STOP was negated (high) at TP3, the timing generator continues to run.	4
	INT STROBE	This 100 ns positive-going pulse usually occurs at TP3 time. If NOT LAST XFER is asserted, INT STROBE does not occur at TP3 time. Instead, TP3 is used by the peripheral to generate BUS STROBE. Timing is suspended until a BUS STROBE is received with NOT LAST XFER high. At that time, INT STROBE is generated and the processor enters TS4.	4
	TS1, TS2, TS3, TS4	These time state lines are high if negated, and low if asserted. Each time state precedes its corresponding time pulse. Time states are always 200 ns or more in duration, and change 50 ns after the leading edge of the time pulse. The machine is in TS1 when stopped.	4
Miscellaneous Signals	RUN	When low, RUN indicates that the machine is executing instructions.	3

Table 9-1 OMNIBUS Signals (cont)

SIGNAL GROUP	SIGNAL	FUNCTION	LOAD
		The Run Flip-Flop is set by Mem Start and cleared at TP3 if STOP is asserted.	
	POWER OK	POWER OK, when high, indicates that the dc voltage from the power supply is adequate to allow proper functioning of the machine. If this line becomes negated, no new memory cycles will be started. Also, after a delay long enough to complete the current cycle, the memory drive current is inhibited. When this line is negated, INITIALIZE is generated.	5
	INI-TIALIZE	INITIALIZE is a positive-going pulse of at least 600 ns duration. This pulse is used to clear AC and LINK, and to clear all flags in peripherals. It is generated if POWER OK is negated, by the Clear Key on the front panel, and by an IOT (6007).	5
	USER MODE	This signal originates in the Time Share portion of the Extended Memory Control. When asserted (ground),it disables OSR, LAS,IOT, and HLT instructions. OSR and LAS are disabled at the panel by inhibiting the placing of SR on the DATA lines. The IOT and HLT instruction are disabled in the Central Processor.	3

DATA TRANSFER, ADDRESSING, AND DATA CONTROL LINES

Data transfer can occur in any one of three data transfer facilities. These are: Programmed I/O Transfers, Interrupt Facility, and Data Break Facility. The means of transferring data requires addressing memory (if data is to be placed into or removed from core). If the data break facility is used, addressing memory is a necessity. If a Programmed I/O Transfer or an Interrupt facility is being used, data must first go to the AC where it is either acted upon or transferred into the Memory Buffer for storage in an addressed memory location. Figure 9-7 illustrates the path of the 12 Memory Data lines, the 12 Data lines, and the 15 Memory Address lines, all of which are on the OMNIBUS. The majority of the control signals that are provided on the OMNIBUS are not shown.

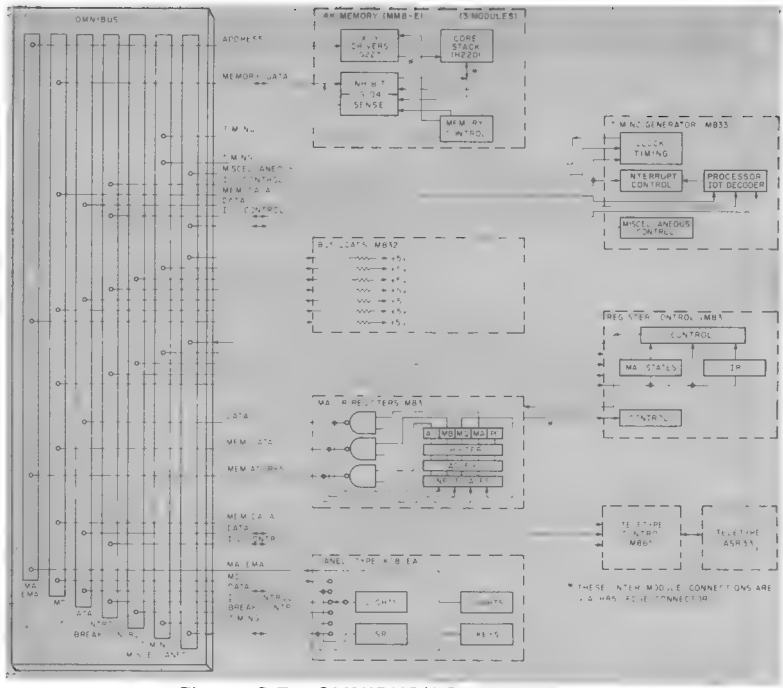


Figure 9-7 OMNIBUS/LOGIC Interface

Data Paths—All data that is to be transferred into or out of a peripheral uses 12 data lines called the DATA lines located on the OMNIBUS. The AC and Interface control are the primary recipients of the data on the DATA lines. For a programmed I/O transfer or an interrupt, data in the AC remains until another instruction acts upon the data. However, Data Break transfers completely bypass the AC. Information from the DATA lines is immediately transferred to the Memory Buffer via the adder circuits and shifters. From the MB register, the data is then transferred to the MD lines and then to the Sense/Inhibit circuits.

Data transfer timing is illustrated in Figure 9-8. During TS1, the operator can observe the status of any one of these data lines on the Programmer's Console. The input data transfer occurs during TS2; the data is loaded into the MB at TP2 time and is then written into memory during TS3 and TS4.

I/O transfers occur during TS3. TP3 is used to generate an AC load signal in the peripheral to load the AC for data transfer from a peripheral.

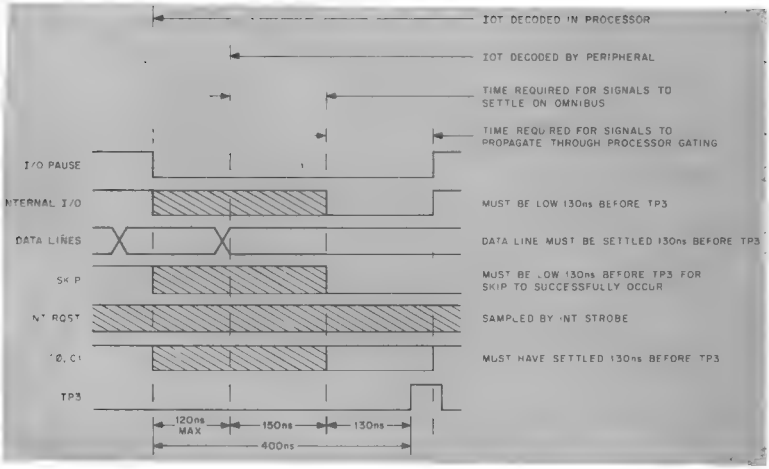
The object of data breaks is to input or output data between memory and the peripheral without disturbing the AC. To input to memory, the DATA lines are used during TS2. Data is transferred into the adder circuits and from there placed into the Memory Buffer, at TP2 time. The contents of the MB are placed into the contents of some memory location. For output transfer, the information is read from memory

and placed into the peripheral control input register via the MD lines.

Memory Addressing—If a peripheral is a programmed transfer device, memory addressing is handled entirely by the program. Data is moved between the PDP-8/E's memory and accumulator by means of the TAD and DCA instructions. IOT instructions are used to identify the state of device flags and to move data between the accumulator and the peripheral's registers. If the device is a data break peripheral, the peripheral controller supplies a 12-bit or 15-bit address to the MA Lines that go into the XY Drivers and Current Source board as the Current address. The source of current address information can be either hardware register (for 1-cycle break) or memory (for 3-cycle break).

Device Addressing—Addressing a programmed transfer device is accomplished as shown in Figure 9-9. If, for instance, the peripheral has data to send to the processor, an interrupt request is initiated because the device flag is set. Upon receipt of the interrupt request, the processor finishes its current instruction, honors the interrupt request and then begins polling all devices (checking all flags) to determine which device needs servicing. Each device address (bits 3-8) is stored in a memory location. The processor begins by pulling each device address out of memory and sends the 6 bits down the MD bus to each peripheral control module device decoder circuit. When a device that has a set flag receives its device address, the flag causes the generation of a skip signal, which then tells the processor that the device has been found.

Thus, as indicated in Figure 9-9, the interface control module only responds to its own assigned device code. Upon receipt of its own device code, a chain of events occurs which includes the assertion of a skip signal, INTERNAL I/O, and the C line.



Programmed I/O
Figure 9-8 Transfer Timing

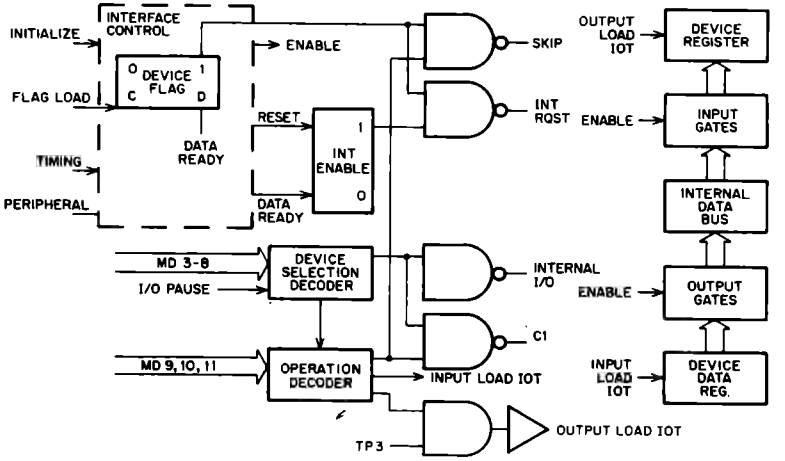


Figure 9-9 Typical Interface Control Module

Data Break Addressing—The data break system is organized on a one-device-per-channel basis. Since each device supplies its own memory address, and since only one device at one time may use the break system, there is no need for device addressing.

Control Signals—Control signals to and from an Interface Control Module (refer to Figure 9-8) ensure that the processor and the peripheral are ready for the data transfer. For example I/O PAUSE is generated by the first 3 bits of the memory word. I/O PAUSE is used as a gating input to the device selection decoder. Because there is the External bus to consider when selecting a device for servicing, an INTERNAL I/O signal is developed by the selected device to tell the external bus interface module to ignore all other signals on the OMNIBUS. When dealing with a device that can both input and output data, it becomes necessary to specify in which direction data is to be transferred on the Data Bus. This is determined by the operation decoder, which then makes C1 either a one or a zero. In the example in Figure 9-9, when C1 is pulled low, data is transferred from the device to the AC register. Timing signals are also used to ensure that both the control signals developed in the interface control module and the data transfer gates are enabled at the proper times with respect to the memory cycle.

DEVICE CODES

The device codes for selecting each device are given in Table 9-2

Table 9-2
PDP-8/E Device Codes

00	Central Processor, Type KK8-E
01, 02	High Speed Tape Reader/Punch & Control, Type PC8-E, PP8-E, PR8-E
03, 04	Console Teleprinter Control, Type KL8-E
05, 07	Oscilloscope Display Control, Type VC8-E, VS8-E
10	Power Fail Detect and Auto Restart, Type KP8-E
10	Memory Parity, Type MP8-E
11	Redundancy Check Control, Type DP8-EP
12	Reserved for Card Punch and Control
13	Real Time Clock, Type DK8-EA, DK8-EC, DK8-EP
14-17	Reserved for Special Systems and Customer's use
20-27	Memory Extension and Time Share Control, Type KM8-E
30-37	Reserved for Asynchronous Data Interface
36, 37	General Purpose Interface, Type BB08
40-47	Synchronous Data Interface, Type DP8-EA, DP8-EB
50-51	
52	Analog Multiplexer, Type AMB-EA
53	A/D Converter, Type AD8-EA
50-57	Buffered Digital I/O, Type DR8-EA
60-62	Disk and Control, Type DF 32D
60-62, 64	Disk Control, Type RS08
63, 67	Card Reader and Control, Type CR8-E, CM8-E
65	Plotter Control, Type XY8-E
66	Line Printer and Control, Type LE8
70-72	Reserved for Industry Standard Magnetic Tape and Control
70-77	Incremental Tape and Control, Type TU22-AE, TU25-AE, TU28-AE, TU28-CE
73-75	Disk File and Control, Type RK8
76-77	DECTape Control, Type TC08

SECTION 2 INTERFACE DESIGN

DESIGNING AN INTERFACE CONTROL MODULE

The basic consideration for the user in designing an interface module is the kind of interface he wants to build and the degree of complexity required. Much of this decision is related to the type of peripheral that he wishes to interface to the PDP-8/E processor.

Using a program to do most of the interfacing is a procedure that offers a great deal of flexibility. Software can handle a great many functions and reduce the complexity of hardware. Most of the time the computer can be used as a low cost controller. The designer must decide how much time he is willing to take from the processor. This decision is similar to that which a user must make in deciding whether to purchase a fast machine such as the PDP-10 or a slower and more inexpensive machine such as the PDP-8/E.

When designing an interface, the user must remember to keep the interface simple. Points to remember are to never use Data Break when

Interrupt can be used and to never use Interrupt when flags can be tested.

Basically, the user has two options in designing his interface. One option is to build an interface module to the external bus, and the other is to build an interface module to the OMNIBUS.

External bus interfacing allows the designer to deal with the wire-wrap system, which by definition is easier for him to alter. The user then does not have to be concerned with the rigid pin assignment imposed upon the OMNIBUS. Chapter 10 explains how to do such interfacing.

Interfacing to the OMNIBUS is simple and direct, providing that the designer conforms to the bus pin assignment. He has several options in selecting the type of module that he wishes to place on the bus. He may, for instance, construct a wire-wrap assembly and place it at the far end of the OMNIBUS. This assembly can be connected directly to the OMNIBUS. There is enough room to place a fairly complicated controller with wire-wrap pins. The restriction is, of course, that the pin assignment must conform to the OMNIBUS. The user may purchase a single quad board containing wire-wrap pins and IC sockets from DEC. Upon this board he may easily construct any type of interface to his specifications.

Another method that provides the highest density of components is the use of an etched board. The user can build blank boards and purchase from DEC most of the necessary IC's, connectors and cables. The advantages of this approach are fully realized when large numbers of duplicate interfaces are to be made. The cost of building an etched board is, of all methods, the lowest—provided that enough interfaces are to be constructed so the designer can recoup his rather high initial engineering costs.

Board Layout

When connecting the +5V supply, the designer should try to split up the runs into three separate parts, and connect each run to a different pin. Then, if there is a short somewhere from +5V to ground, it is three times as easy to find. A good rule to follow is to limit the number of IC's to be mounted on any one board to 50. If the designer has a requirement for more than 50 IC's, he should consider using a second board. A typical layout of components is shown in Figure 9-10.

Uninsulated components should not project more than $\frac{3}{8}$ in. above the board, insulated components should not project more than $\frac{11}{32}$ in. above the board. The grounding scheme should be carefully planned. Pins C, F, N, and T (except AC1) provide the ground lines needed to operate the board. It is good practice to use as many ground lines as possible; connect all ground pins together, and make runs as short as possible to the ground pins of the ICs. Poor grounding can cause occasional annoying malfunctions.

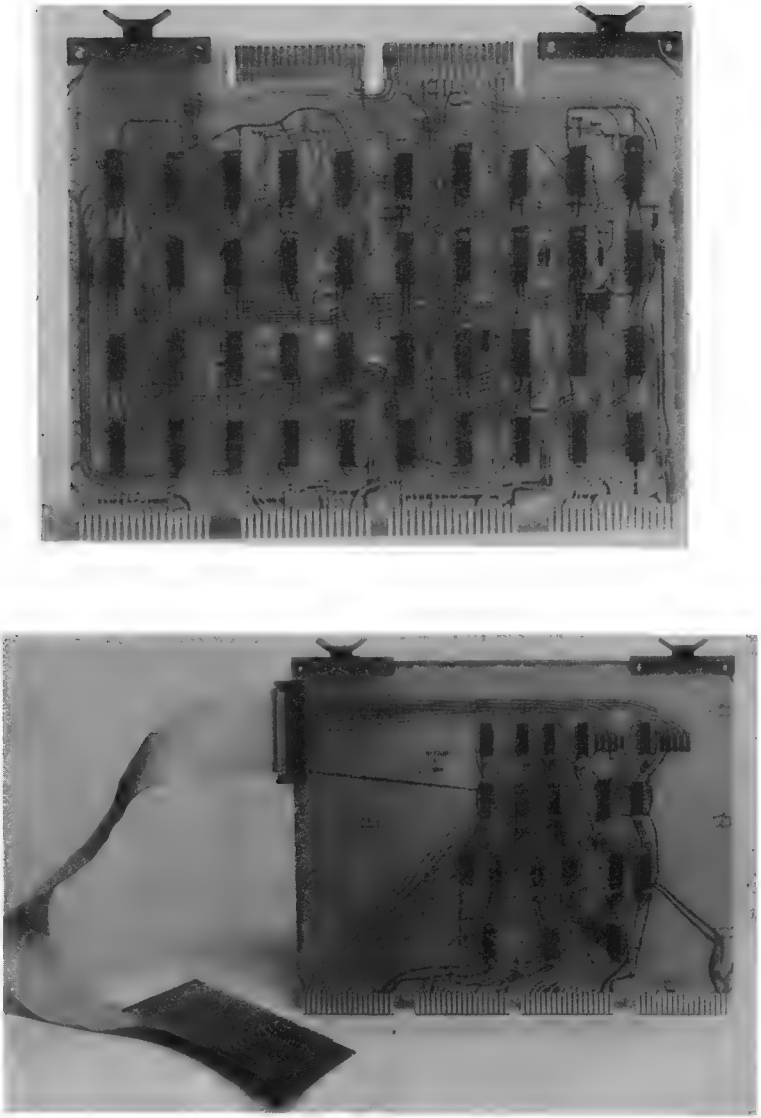


Figure 9-10 Typical Component Layout

Etched Circuit Layout and Construction Rules
The following layout and construction rules should be used as a guide to assure optimum performance of the interface module:

1. Keep all leads short. Leads connecting to the OMNIBUS should be 6" or less in length.
2. Observe IC manufacturer's rules regarding bypassing of power supply and treatment of unused gate inputs.
3. Do not rely on "typical" figures published by IC manufacturers. Next year these numbers may be changed.
4. All lines connected to switches or relay contacts should have a switch filter at the input to the board.
5. If switches are used to generate pulses, use a flip-flop or Schmitt trigger to obtain reasonably fast rise time and absence of contact bounce.
6. Make all ICs point the same way.
7. Timing elements are invariably more sensitive to noise. Try to physically and electrically isolate clocks and delays, and arrange bypassing so that these elements will not be susceptible to +5V noise and other voltages.
8. Test the resistance to ground from +5V, +15V and −15V and other voltages before applying power to the board. If the resistance is less than 10 to 20 ohms, find and remove the short. This precaution may prevent irreparable damage to the board.

Designing Basic Interface Control Circuits

The basic interface control circuits to either transfer data in or transfer data out consists of: 1) a device selection circuit, 2) a device operation circuit, 3) data transfer circuits, and 4) miscellaneous control gates. All of these circuits combined are illustrated on Figure 9-11. A general rule to follow is that signals that are taken off the OMNIBUS are done so with either Signetics SP380, SP314, or SP384 ICs (or equivalent). This is recommended to minimize bus loading. Signals placed on the OMNIBUS should be gated with Signetics 8881 ICs (or equivalent). This is recommended because of the low leakage current characteristic that allows a greater number of devices to be "wired ORed" together.

Device Selection Circuit—MD 3-8 bits are used to carry the device select information. The example given in Figure 9-11 shows SP380 and SP314 being used as a simple decoder. When octal 57 is received, the INTERNAL I/O signal to the bus is pulled low and device operation bits MD 9, 10, and 11 are gated into the binary to octal decoder. I/O PAUSE must be pulled low by the central processor before device selection is possible.

Device Operation Circuit—MD 9-11 bits determine the type of operation to be performed. Three SP380's are shown receiving these bits. The outputs of these gates are in turn presented to a binary-to-octal decoder. The resulting IOTs are used to control the type of transfer or reset the flip-flops after data transfer is completed.

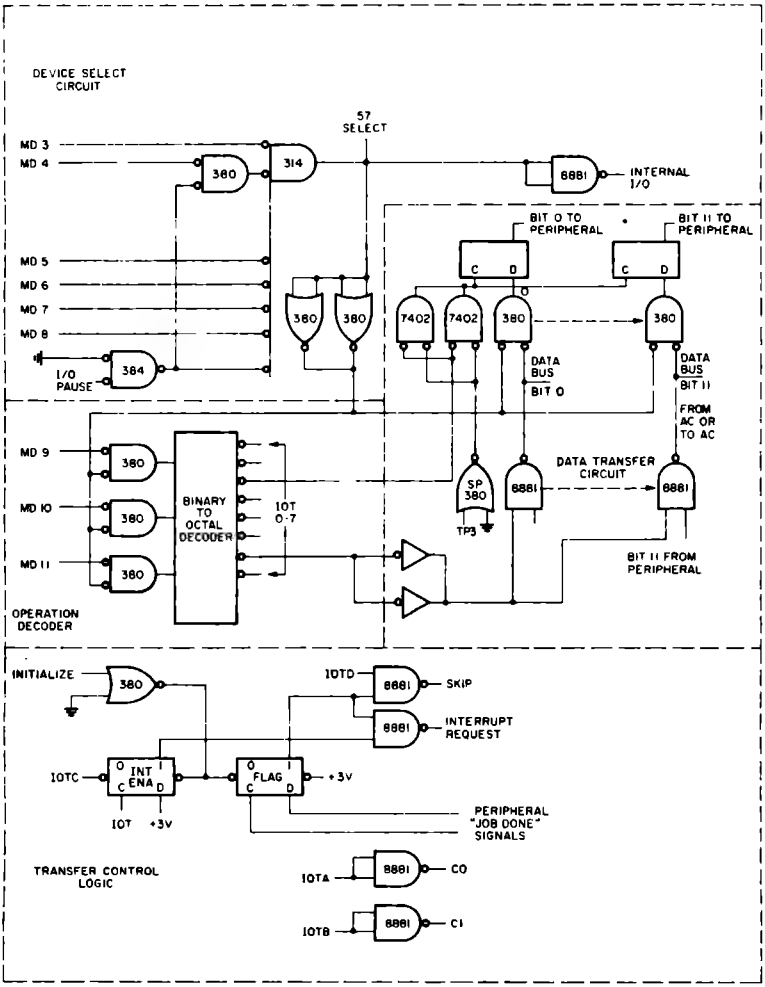


Figure 9-11 Basic Interface Control Circuits

Data Transfer Circuit—To transfer all 12 data bits on the data bus, there must be 12 parallel chains of logic similar to those illustrated. A buffer register is shown as the output circuit to the peripheral data bus. Each data line connects to a 380 IC, and the data on each line is gated in by the select input to each of the 12 gates. The flip-flop is clocked at TP3.

To receive data from the peripheral, each data line is placed on the input of an 8881 IC. An IOT from the binary-to-octal decoder is used to gate the information onto the DATA lines to the OMNIBUS.

Transfer Control Logic—Both SKP and INTERRUPT REQUEST operate from the Interrupt Enable and Flag flip-flops. The flag is usually set by a completion signal from the peripheral. Signals C0 and C1 are necessary to indicate to the processor the type of transfer.

All signals which load flip-flops or trigger delays in the peripheral should be "ANDED" with TP3. Programmed I/O signals which are inputs to the processor are generally gated with a pulse within the processor.

General Cable Rules and Suggestions

Cabling is an important consideration when designing and constructing an interface control module that plugs into the OMNIBUS. The designer's only concern is that his cabling is adequate between his peripheral and the interface control module that he is designing to plug into the OMNIBUS. No additional cabling is required. The interface cable connects to a 40-pin connector type H854 shown mounted on the upper left side of the module. The peripheral connector is a 40-pin type connector. A channel "cut-away" along the length of the power supply allows bundling of many cables and allows them to be clamped onto the side of the channel.

DEC Supplied Interface Cables

The user may purchase from DEC the necessary interface cables. Two standard lengths are provided—a six ft. cable (part number BC08J-6) and a ten ft. cable (part number BC08J-10). Each cable contains a 40 pin connector type H855, which connects to the interface module and a 36 pin module type M953 which in turn connects to the users peripheral.

Each cable provides 18 signal lines and 22 ground lines. A ground line follows each signal line with several ground lines on each end of the cable. This arrangement is illustrated in Figure 9-12.

The design is intended to provide an electrical shield between lines, and to provide adequate grounding between the two ends of the cable. The use of ribbon cable is convenient, saving space and eliminating bundle problems. However, round coaxial cabling can be used. The coaxial cable then has to be larger because it consists of several layers of conductor and insulator material. Where the environment is considered hostile (such as may be the case in a factory), coaxial cabling is recommended. However, in most cases, the flat cable will prove adequate.

Module M953 Pins		Module H855 Pins	
Signal	Ground	Signal	Ground
	A1		UU, WW
B1		TT	
	C1		PP
D1		NN	
E1		JJ	
	F1		EE
H1		DD	
J1		Z	
	K1		W
L1		V	
M1		R	
	N1		M
P1		L	
	R1		H, K
S1		F	
	T1		
*U1			
*V1			
*A2			
*B2			
	C2		SS
D2		RR	
E2		LL	
	F2		HH, KK, MM
H2		FF	
	J2		CC
K2		BB	
	L2		Y, AA
M2		X	
	N2		U
P2		T	
	R2		P, S
S2		N	
T2		J	
	U2		A, B, C, E
V2		D	
* not used			

Figure 9-12 Interface Cable Pin Assignment

Cabling Rules

There are cabling precautions that the designer of any interface control module should follow. These include:

a. Do not run the AC line immediately adjacent to a low level signal.

b. Do not attempt to drive a line directly with the output of a flip-flop. The flip-flop may be triggered by noise being sent back along the line.
c. Tie all grounds together at the board and at the far end of the cable.
d. If there are two or more cables running parallel, there must be an intervening electrostatic shield.
e. Use as low an impedance as possible, but not lower than 100 ohms on the lines; and terminate the lines at the far end to eliminate ring, or drive them with a higher impedance and wait for them to settle down.

If the user requires a more complicated interface controller that requires two boards, a connector type H851 is used to interconnect the signals from one board to the next. This is shown in Figure 9-2. The connector receives 36 etched finger type pins from both modules and slides onto both modules. Pin A1 connects to pin A1, etc.

INTERFACE TIMING CRITERIA

In nearly all instances, the user need not concern himself with the details of timing. This section is included to assist the person with an exceptional case.

There are basically three types of timing with which the user may be concerned. First, there is the data exchange time between the computer itself and the peripheral (a function of the I/O structure of the machine). Secondly, there is data break timing, which is a function of the break priority system and the data break peripherals. The third consideration is Interrupt timing.

General Timing Rules

General timing rules are as follows:

a. If maximum machine speed is necessary, do not use the positive I/O Bus interface, type KA8-E. Instead, design all peripherals so that they plug directly into the OMNIBUS.
b. When the above is not feasible, do not microcode IOTs. For example, replace the KRB instruction with its equivalent KCC and KRS instruction. The result is a slightly longer time for the overall IOT; however, the processor is stopped for two shorter periods of time, rather than one long period of time. Hence, the data break and interrupt systems have access to the processor and memory sooner.

Interrupt Timing

Interrupt timing is concerned with the interval from the time that a flag is seen until the time it can be serviced. Although this time is fairly easily calculated, the exact details depend upon the number of flags the processor checks before it finds a flag that is set.

Timing Example:

An interrupt request is asserted by one of the control modules. The processor must finish the current instruction upon which it is operating, possibly a TAD indirect through an auto index register. TAD indirect

through an auto index register takes 1.2 microseconds plus 2 times 1.4 microseconds, or 4.0 microseconds total. In addition, the processor might not have seen the flag as much as 300 manoseconds before that time, so it is a maximum of 4.3 microseconds from the time that the flag got set until the time the processor can start executing the JMS to location zero. This assumes no EAE option and that no device is using the break facility at that time. Thus, up to this time, it has taken the processor 4.3 microseconds to recognize that there is a flag to be serviced. This example assumes that the KM8E Memory Extension control is installed.

TIMING EXAMPLE

TAD I	4.0 μs	Through an Auto Index register
	0.3 μs	Worst case time for processor to see flag
(JMS)	1.4 μs	Jump to location 0, store PC
DCA AC	2.6 μs	Store AC and Link
GTF	1.2 μs	Pick up flags in memory extension control unit
DCA FLAGS	2.6 μs	Store flags
MQA	1.2 μs	Load AC with MQ
DCA MQ	2.6 μs	Store MQ
JMP I . + 1	2.2 μs	Jump somewhere to Interrupt flag scan
IOT FLAG SKIP	1.2 μs	Skip on flag—Start of Flag Scan Routine
IOT	1.2 μs	Perform I/O Transfer
Total	20.5 μs	Total time required to get to IOT and perform instruction.
DCA I 10	4.0 μs	DCA indirect through some auto index register
CLR FLAG	1.2 μs	
ISZ	2.6 μs	Bookkeeping—To see if last transfer
JMP	1.2 μs	Dismiss
TAD MQ	2.6 μs	Restore MQ
MQL	1.2 μs	Transfer the contents of AC into MQ
TAD FLAGS	2.6 μs	Check status of flags
RTF	1.2 μs	Restore flags
TAD AC	2.6 μs	Restore AC
JMP I 0	2.4 μs	Go back to location 0
Total	21.6 μs	Total time required to get back to main program again

Thus 20.5 μs + 21.6 μs = 42.1 μs total for this example.

The interrupt timing requirements are a function of the amount of coding that it takes to determine what has to be done and how much time is available to do it. There are two times that must always be considered. The first is the length of time from the time of the interrupt request until the source of the interrupt request has been recognized and the data retrieved. The second time is the length of time it takes to finish

processing the data, including all of the housekeeping routines and the time it takes to restore the major registers to their original state.

A special instruction, SRQ (octal 6003), allows the programmer to test for possible interrupts before actually restoring the machine. One may save considerable time entering and exiting the interrupt program by merely returning to the flag scan routine, if the SRQ instruction indicates the presence of a second interrupt.

The program interrupt system is satisfactory for data rates less than 10 KHz (one word every 100 microseconds). Above this rate, the user should examine each individual case for its merits and decide whether or not he should use the data break facility or possibly tie the machine up with high data rates.

Timing Requirements for Data Break Facilities

The important timing consideration in data break as in program interrupt, is whether sufficient time is available from the time the flag (in this case, the break request) is set to the time all the data is moved in or out of memory. The first item in question is the period of time before operation actually starts on the break request. The break request timing for the PDP-8/E has improved considerably in that a break request can be honored between major states of an instruction, whereas in the older model machines, the break system had to wait until an instruction was completely processed. The break system is synchronized 300 nanoseconds before the end of every memory cycle. At the same time the processor tests for the possibility of interrupt, it tests for the possibility of break. Unlike the case of an interrupt, the break system need not wait until the processor is ready to go into a FETCH. If the processor is programmed to do only machine instructions (assuming no EAE or external I/O), it would take the processor no more than 1.7 microseconds to begin servicing the break request. For the external bus, this time would be 4.9 microseconds, maximum. The choice of one-cycle or three-cycle breaks is an important timing consideration. Basically, the internal operation of the PDP-8/E is such that it readily adapts to one cycle break. Users who are planning on constructing their own break device should think in terms of one cycle break. However, considerably more hardware is required to build a one-cycle break interface.

Timing and Break Priorities—Timing problems result when the break priority system has not been carefully planned. Suppose, for example, the highest speed break device is a complex parallel disk that serves up a word once every 5 microseconds. Suppose two break devices request a break simultaneously (e.g., the disk and a slower device such as the DECtape that has 50 microseconds for which the word is available). If the priorities of these two devices are not correct (i.e., DECtape assigned first priority), timing problems are inevitable. If the DECtape and disk simultaneously request a break, both devices must wait 1.7 microseconds for the current memory cycle to finish. The DECtape then makes a 3-cycle break request since it (incorrectly) has the highest priority.

Therefore, the waiting time of the disk is 3 times 1.4 microseconds or 4.2 microseconds plus 1.7 microseconds, (a total of 5.9 microseconds), to service the DECtape. The result is that the DECtape was serviced and

the disc has lost its data because 5 microseconds have expired. This is an obvious situation where the disk must have a higher priority. Other situations may not be obvious without examining the timing requirements before assigning a priority to each device. As a general rule, the user should set up his priority based upon the device that has data available for the shortest amount of time.

GENERAL PROPAGATION DELAY GUIDELINES

When designing an interface module, the designer should consider the individual propagation delays of such logic elements as NAND gates, flip-flops, J-K flip-flops, one shot delays, etc. He should add each delay in a logic chain to determine the overall delay of his module.

2 Input NAND Gate Delay

Typical characteristics of a NAND gate used with the PDP-8/E logic are illustrated in Figure 9-13. Where high fan out is required, a SN7440 type gate is preferred.

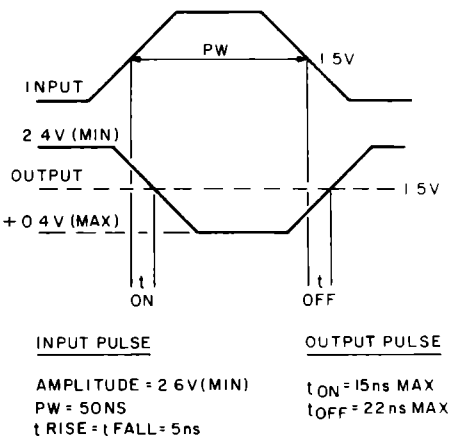


Figure 9-13 2 Input NAND Gate Typical Characteristics Example

Flip-Flop Propagation Delays

Typical D-type flip-flops trigger on the leading or rising edge of a positive clock pulse; the propagation delay is measured from the threshold point of this edge. The set-up time of the flop is also measured from this threshold point. Data on the input must be settled at least 20 nanoseconds prior to the clock transition.

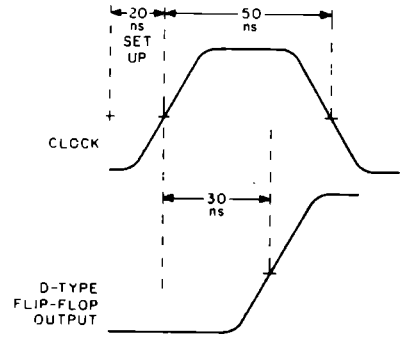


Figure 9-14 Typical D-Type Flip-Flop Timing Example

J-K flip-flops

J-K type flip-flops are, in effect, trailing edge triggering devices as explained previously. The only restriction on the J and K inputs is that they must be settled by the time that the rising edge occurs. Timing is shown in Figure 9-15.

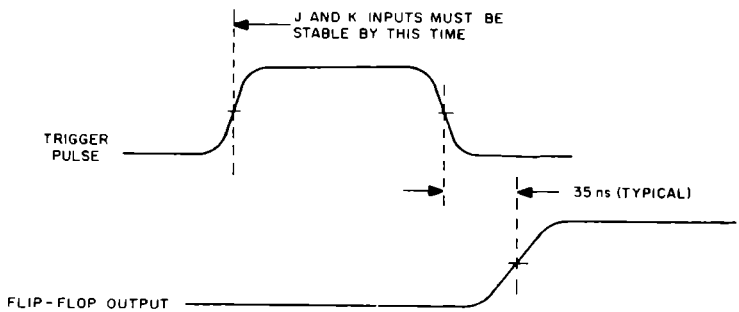


Figure 9-15 J-K Flip-Flop Timing Example

When using the dc Set or Reset inputs of either flip-flop type, propagation delays are referenced to the falling edge of the pulse. This is due to the inverted sense of these inputs. When resetting ripple type counters (where the output of one flip-flop is used as the trigger input to the next stage), the Reset pulse must be longer than the maximum propagation delay of a single stage. This ensures that a slow flip-flop does not introduce a false transition, which could ripple through and result in an erroneous count.

One-Shot Delay—Calibrated time delays of adjustable duration are generated by the Delay Multivibrator such as the M302. When triggered by a level change from a logical one to a logical zero, this module produces

a positive output pulse that is adjustable in duration from 50 to 750 ns with no added capacitance. Delays up to 7.5 milliseconds are possible without external capacitance. Basic timing and the logic symbol are shown in Figure 9-16. The 100 picofarad internal capacitance produces a recovery time of 30 ns. Recovery time with additional capacitance can be calculated using the formula:

$$t(r)\ \text{Nanoseconds} = 100.3\ C\ \text{Total (Picofarads)}$$

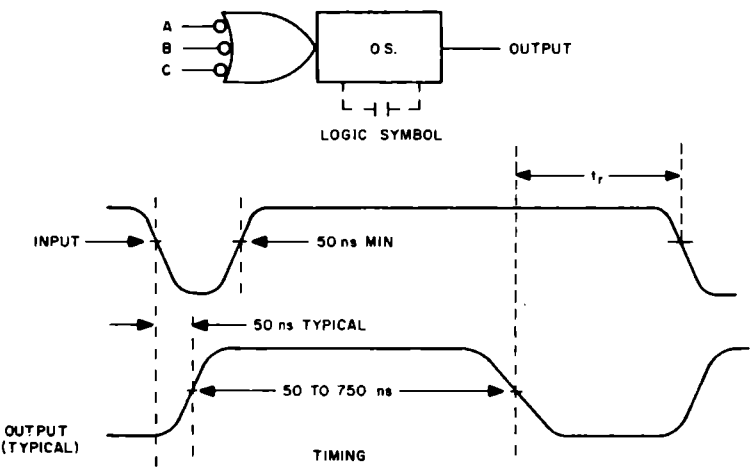


Figure 9-16 One-Shot Delay Timing and Logic Symbol Example

Maximum Operating Frequency

Once the designer has determined the individual propagation delays in each logic element, he must then add these delays corresponding to a simple logic chain. He then compares the results with the system frequency to assure that his logic circuit can meet the requirements imposed by the system frequency. Figure 9-17 illustrates a situation in which various logic components in a given chain are examined and all delays are added. The following assumptions are made:

a. A standard clock pulse width of 50 nanoseconds is assumed. This period is measured from the threshold point of the leading edge to the threshold point of the trailing edge.
b. One flip-flop propagation delay of 35 nanoseconds from the trailing edge of the clock pulse to the threshold point of the final state of the flip-flop is allowed.
c. Two gate-pair delays of 30 ns each are assumed. (A gate-pair consists of two inverting gates in series.) Two gate-pair delays are usually required to perform a significant logic function with a minimum of parallel operations. The two gate-pair delays total 60 ns.

The time necessary to perform these operations before the next occurrence of the clock pulse is the sum of the delays; 50 + 35 + 60, or 145 nanoseconds. Allowing 20 ns for variations within the system, the resulting period is 165 ns, corresponding to a 6 MHz clock rate.

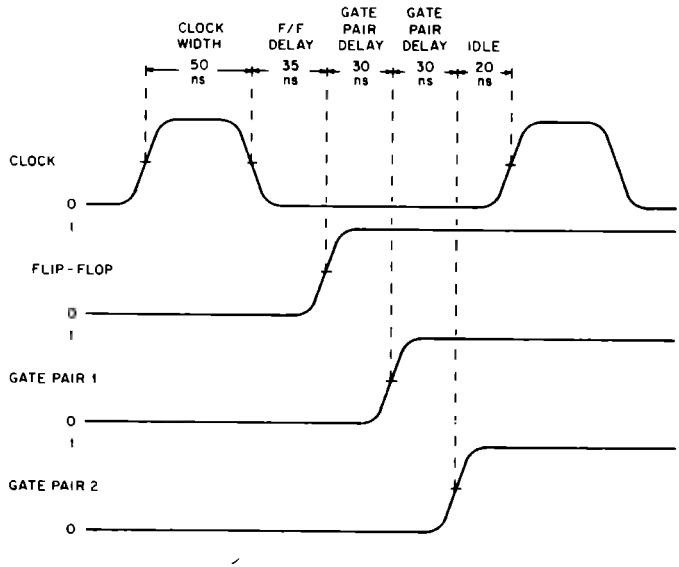


Figure 9-17 Delays Determining Total Operating Frequency Example

Note that the D-type flip-flop triggers on the leading edge of the clock pulse and the J-K flip-flop triggers on the trailing edge. When calculating system timing using flip-flops, remember that the flip-flop inputs must be settled at least 20 nanoseconds prior to the occurrence of the clock pulse.

The preparation of a timing diagram that considers delays introduced by all logic elements aids the designer in achieving predictable system performance. Don't forget that minimum (including zero) delays are possible, and that no good design should rely on a finite gate delay.

Using a similar approach, the designer must ensure that his selection and operation decoding scheme works within the allotted time, and that data can be made available in time for TP3 to either strobe the data in or strobe the data out, depending upon the type of transfer. A simple timing diagram illustrating this activity is shown in Figure 9-8. I/O PAUSE is used to gate the device and operation decoders and TP3 strobes the data either into the OMNIBUS or onto the data lines to the peripheral. A total of 300 nanoseconds is allowed to bring the contents of either the input or output data line into the interface controller. All input signals to the processor must be established at the processor at

least 130 nanoseconds before TP3. This discussion assumes either an I/O transfer or a programmed Interrupt. With data break devices, the preceding information does not apply.

LOADING RULES

Almost all signal lines on the OMNIBUS are driven positive by a load resistor (refer to Figure 9-6) and pulled to ground by open collector ICs. The designer should use a gate such as the Signetics SP380, SP314, SP384 ICs or their equivalent as an input device. These gates have a fairly high input impedance and, therefore, do not load down the bus. Wherever possible, one other input gate should be held positive by I/O PAUSE or some other logic signal in order to further reduce the input load. Conversely, to drive output lines (such as the bus lines) the user should consider Signetics 8881 or its equivalent (paying careful attention to the leakage current.)

Device Selection Inputs

When designing the device selection decoder, a NAND gate such as the SP380, which is a two-input gate, should be strobed by I/O PAUSE. This helps to remove the load from the MD lines.

Skip and Interrupt Request Lines

As a general rule, the Signetics 8881 gate outputs should be limited to one per device code for skip line and one per device code for the Interrupt Request Line. A potential problem exists when too many gates are tied into these lines.

Electrical Considerations of Driving a Line

Most signal lines on the OMNIBUS are tied through a load register to +5V. Users who want to look at any one line must do so with the Signetics SP380 gate. Users desiring to drive any bus line to ground must do so with the output of the Signetics 8881 gate. The limitation of the amount this gate can carry must be considered. A major factor is the output leakage, as no switch is a perfect open circuit nor a perfect short circuit. This is a fundamental limitation. Tolerance for ground level should be considered up to .4V, as defined by the TTL logic. The high signal must be a minimum of 2.6V; however, 3.0V is recommended. An additional consideration is that the SP380 requires some current at its input.

GROUNDING

Pins C,F,N, and T on the OMNIBUS are used for ground signals. The user who is making modules that are designed to plug into the OMNIBUS should utilize all of the ground pins and tie all of the ground pins together. He should make the connecting lines as short as possible. The user should also attempt to keep the leads from the ground pins to the ground terminals on the ICs as short as possible. The shorter the ground runs from the integrated circuits on the module to the ground pins and the more duplication there is (parallel paths), the quieter the ground system is within the module. The designer should pay careful attention to the recommendations of the ICs to ensure that good construction practices are followed (Do not overlook the local bypass capacitors required at an IC). The designer should use as much as possible a .01 microfrad ceramic disk capacitor across Vcc to ground for every IC used.

TESTING TECHNIQUES

When the interface module has been completely assembled and checked, the designer should perform an initial checkout and then proceed to test his interface in the system. This should be followed by a complete peripheral system integration.

Initial Checkout

The tester should remove the power source connections prior to performing his initial test. He should then make an electrical test with an ohmmeter from +5V to ground, from −15V to ground, and from +15V to ground. He must ensure that there are no power shorts prior to connecting power. A short can damage the etched circuits.

System Test

The next test step is to plug the module into the OMNIBUS and generate an IOT to check out the device selection capability, logic levels, operation of the flag, and capability of the interface controller to receive and transfer data. A combination of 1's and 0's may be placed in the AC and transferred to the interface. With an oscilloscope or voltmeter, the tester can check each of the connector terminals corresponding to each of the data bits to ensure that the right information is being transferred.

Another useful test is to generate a count pattern in the AC and observe that bit 11 is moving twice as fast as bit 10, which is moving twice as fast as bit 9, and so on. These waveforms can be examined for each line. This test indicates shorts existing between data lines.

Final Testing

Before actual operation, the final test includes connecting the peripheral to the interface and transferring data ino and out of the peripheral. For example, the 1's and 0's can be checked at the peripheral end for input to the peripheral and checked at the AC register for data transferred from the peripheral to the processor. Whenever possible, the programs used for testing should be collected into a Diagnostic Program for the device. A properly designed diagnostic, since it tests only one peripheral, is a powerful tool for finding system failures.

PROGRAMMING RULES

The most successful method of programming is to begin a program as simply as possible, test it, and then add to the program until it performs the required job. Before beginning the programming, the programmer should become familiar with the programs that he will be using. Refer to Chapter 4 for a description of the standard programs and refer to Appendix A for a complete list of the PDP-8/E compatible programs.

For best results, the programmer should avoid the use of the following device codes.

1. Device code 0 (reseryed for processor)
2. Device code 3
3. Device code 4
4. Avoid all codes in the 20 through 27 series (reserved for the extended memory control)
5. Avoid the Disk and DECtape device code series

Device codes 14-17 have been made available for the programmer's special use.

APPLICATION EXAMPLE

An example of a common interface control module is given in Figure 9-18. This illustration represents the necessary control logic for a high-speed paper tape punch. However, the example could be modified to accommodate other functions (such as controlling equipment, process, etc).

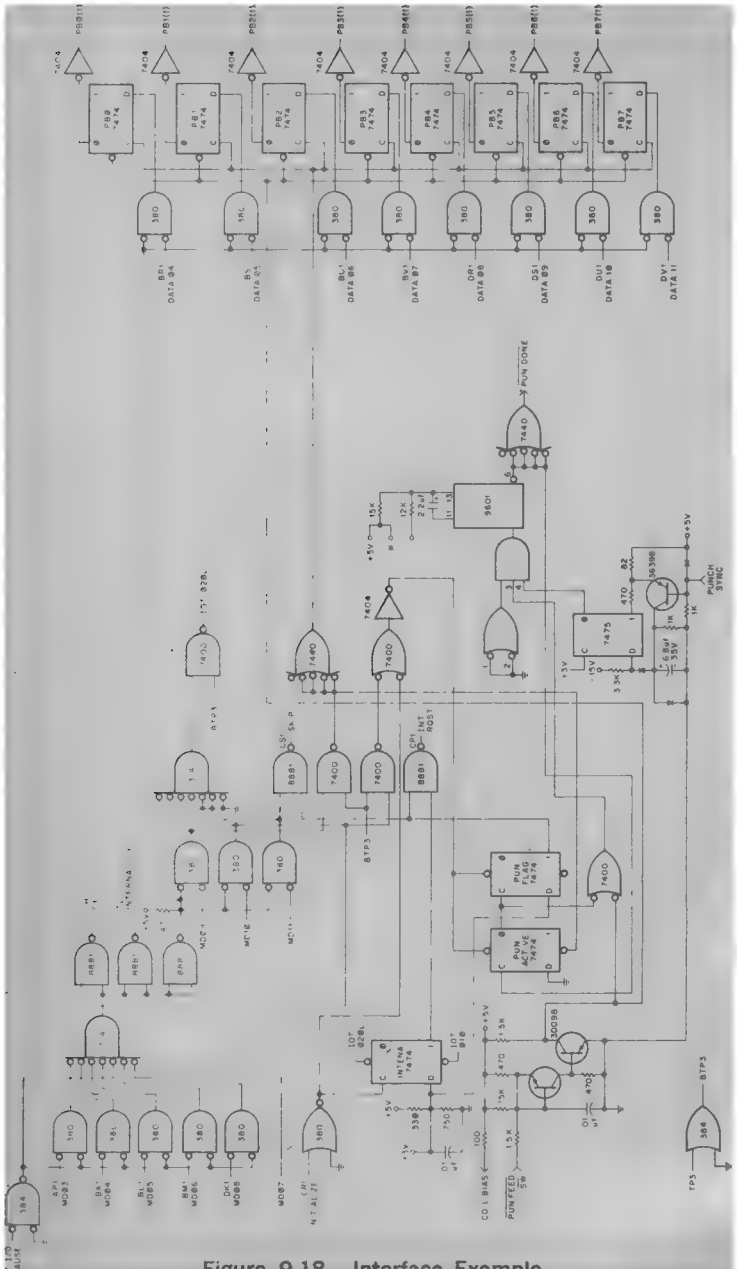


Figure 9-18 Interface Example

CHAPTER 10

EXTERNAL BUS INTERFACING

GENERAL

The degree of versatility of a computer is determined by the type and number of peripheral devices that can be interfaced with it. The PDP-8/E was designed with this idea in mind. Consequently, it can be easily interfaced with a variety of peripherals. Previous discussions have brought out the fact that peripherals can be interfaced with either the OMNIBUS or the external bus. This means that a PDP-8/E user can utilize not only devices designed exclusively for the PDP-8/E, but also devices originally designed for use with the PDP-8/I and PDP-8/L computers, and even devices of his own manufacture.

This chapter deals with the external bus, its applications, and the technique of interfacing peripherals to the bus. The user may wish to interface a DF32-D Disk File and Control unit with the PDP-8/E, for instance. This equipment was designed for the PDP-8/I and PDP-8/L; but, by interfacing to the external bus, it can also be used with the PDP-8/E. The user may want to transfer data between the PDP-8/E and a remote location. The external bus, which is designed to drive long interconnecting lines, is ideally suited for this application.

The first part of the chapter answers general questions about the external bus—What is it? How does it differ from the OMNIBUS? How does one use it? The remainder of the chapter guides the user through the initial uncertainties of interfacing by covering such topics as: types of connectors and cables to use with the external bus; timing criteria, loading rules, and voltage levels; and hardware and wiring techniques. DEC hopes that this information will be helpful in designing and implementing any interface that the user might require.

Should questions arise regarding computer interface characteristics, the design of interfaces using DEC modules, or installation planning, customers are invited to telephone any of the DEC sales offices or the main plant in Maynard, Massachusetts. Digital Equipment Corporation makes no representation that the interconnection of its circuit modules in the manner described herein will not infringe on existing or future patent rights. Nor do the descriptions contained herein imply the granting of licenses to use, manufacture, or sell equipment constructed in accordance therewith.

THE NATURE OF THE EXTERNAL BUS

What is the external bus? It is simply a number of signal lines (88, excluding grounds) that enable data transfers between the CP and peripherals. These lines carry data and control signals between the peripheral and two interface boards—the Positive I/O Bus interface (KA8-E) and the Data Break interface (KD8-E)—that plug into the PDP-8/E OMNIBUS. These two boards convert the internal bus signals into PDP-8/I and PDP-8/L-type bus signals. For instance, PDP-8/I peripherals need IOP pulses to perform instructions. The PDP-8/E does

not generate internal IOP pulses, but it does provide signals (MD bits 09, 10, and 11) that can be converted into IOP pulses by the Positive I/O Bus interface. Other signals normally required by these peripherals are, in essence, available on the OMNIBUS. For example, BAC (buffered accumulator) bits must be supplied for the PDP-8/I peripherals. The PDP-8/E Data lines carry the necessary accumulator information. The Positive I/O Bus interface merely buffers the DATA bits and, thus, provides the external bus BAC signals.

Although the external bus consists of signal lines from both the positive I/O Bus interface and the Data Break interface, it is not always necessary to use both boards. When only programmed I/O transfer peripherals are used, the Positive I/O Bus interface provides all the necessary signals. However, if data break peripherals are to be connected, both interfaces must be used. Because each data break peripheral requires its own data break interface board, the number of signal lines comprising the bus may vary. There may be as many as 12 of these data break peripherals connected in the system, each contributing 36 signal lines to the external bus. Figure 10-1 illustrates the bus and its use when applied to a series of peripherals.

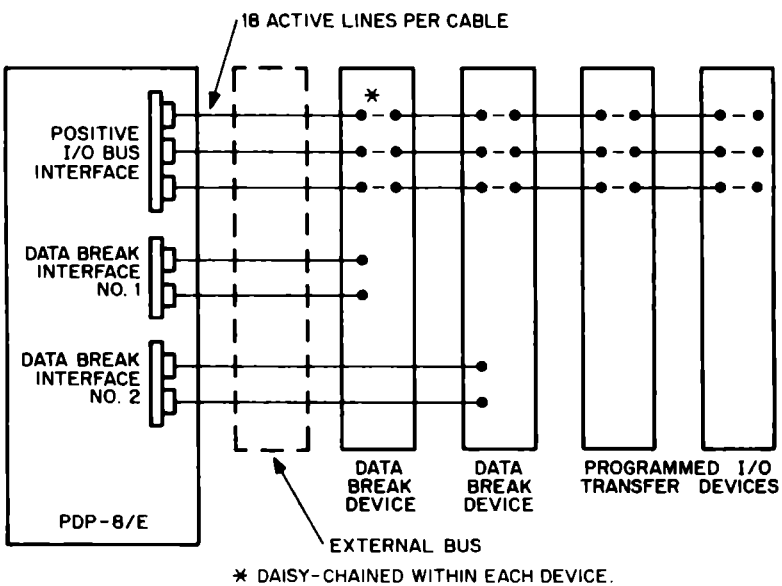


Figure 10-1 Parallel Connection of Peripherals

EXTERNAL BUS SIGNALS

Figure 10-2 shows not only the external bus signals, but also those OMNIBUS signals that are used by the two interfaces. Signal directions are shown for both buses. Some of the OMNIBUS signals—DATA 0-11,

for instance—are common to both interfaces, but for clarity this commonality has been disregarded. The external bus signals are grouped according to the interface connector where they originate (Table 10-1 in paragraph 10.5 lists the bus signals and the connector and pin where each may be found). When similar signal lines are represented by one line of the drawing, as BAC 00-11, the actual number of lines is indicated in parentheses. The external bus signals are discussed in detail in the following section with emphasis on the relationship between these signals and the OMNIBUS signals.

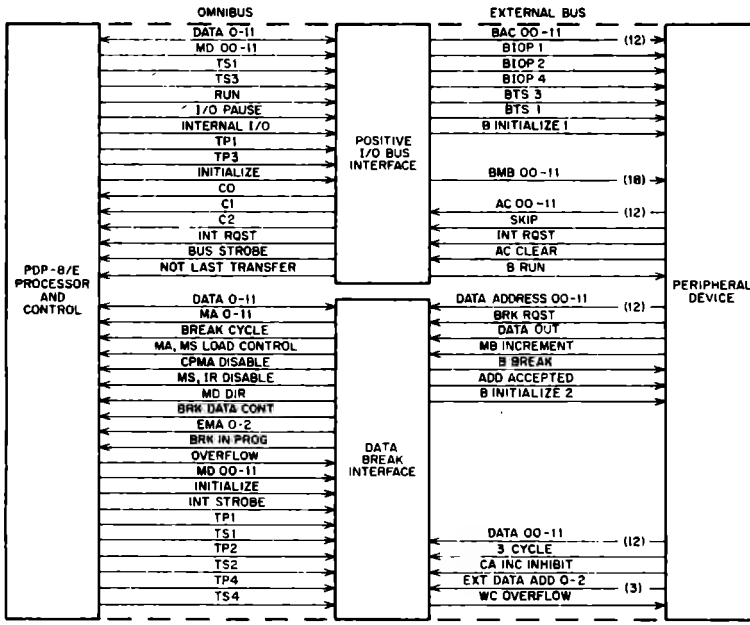


Figure 10-2 External Bus Signals and Related OMNIBUS Signals

SIGNAL NAME	DESCRIPTION
BAC 00-11	These signals represent the content of the PDP-8/E Accumulator register (AC). Information in the AC is transferred on the OMNIBUS DATA lines to the Positive I/O Bus interface. The interface buffers the signal and provides the BAC output. The BAC bits are strobed into registers in the peripheral when an IOT instruction is generated. 1 = +3V.
AC 00-11	The signals on these lines represent the contents of a register in the peripheral. This information is transferred to the Positive I/O Bus interface where

SIGNAL NAME	DESCRIPTION
	it is put on the data lines for transfer to the PDP-8/E AC. 1 = GND.
BMB 00-11	The signals on these lines represent the content of the Memory Sense registers. This information is transferred from memory on the Memory Data (MD) lines. The MD lines are monitored by the Positive I/O Bus interface and the signals are converted to the BMB bits. These bits are used during IOT instructions; BMB03-08 carry the device selection code, while BMB09-11 are converted to BIOP pulses. 1 = +3V.
BIOP 1, 2, & 4	These pulses are generated in response to the voltage levels on MD09-11 (BIOP4-1, respectively). These pulses generate IOT pulses within the peripheral, causing it to perform a certain operation. The width of the BIOP pulses and the interval between pulses are variable and can be adjusted on the Positive I/O Bus interface. Pulse = +3V.
BTS1, BTS3	These signals represent the TS1 and TS3 signals of the OMNIBUS. They synchronize operations in the peripheral with those in the computer and perform functions peculiar to the peripheral. They are primarily used in data break timing. Pulse = +3V.
B RUN	If this signal is GND, the computer is executing instructions.
AC CLEAR	When this signal is asserted (brought to GND) along with the AC bits, the result is a jam transfer of data to the AC. The signal may also be asserted by a separate IOP, clearing the AC.
SKIP	SKIP is asserted (grounded) by an IOT instruction. It causes the next sequential instruction to be skipped. If the SKIP bus is asserted during more than one IOT of an I/O instruction, the program skips a corresponding number of instructions. No more than three skips can be made by a single instruction.
B INITIALIZE 1	This 600 nanosecond-duration positive pulse is used to clear AC and link and to clear all flags in peripherals. It is generated at power turn on, and by the Clear All Flags (CAF) IOT, 6007.
DATA 00-11*	These lines transfer data from a data break peripheral to the data break interface. The peripheral transfers the information when it receives the B BREAK signal from the interface, indicating the start of the true break cycle. At TS2 of this break cycle, the data

SIGNAL NAME	DESCRIPTION
	break interface transfers the data to the OMNIBUS DATA 0-11 lines, which carry the data to the CP's memory buffer. 1 = GND.
B BREAK*	This signal is generated in the data break interface and transferred to the peripheral, where it enables a parallel loading of data, either into or out of the peripheral. The data break interface, in addition to generating B BREAK, asserts the OMNIBUS BREAK CYCLE line, notifying the computer that the break cycle has begun. 1 = GND.
DATA OUT*	This signal is produced by the peripheral and sampled by the data break interface. When DATA OUT is asserted (grounded) during the break cycle, data is transferred from the computer's memory to the peripheral.
DATA ADD 00-11*	These lines transfer address information from the peripheral to the OMNIBUS MA lines. If the peripheral is a 3-cycle break device, the address represents the memory location of the word count. Since this location is always the same for a 3-cycle device, the DATA ADDRESS lines are hard-wired in the peripheral. This address must be even (ending in 0, 2, 4, or 6) for word count. The data stored in this location represents the 2's complement of the number of data words to be transferred. The next sequential location is read from memory as the Current Address register. The data stored in this location represents the memory address of the data to be transferred. If the peripheral is a 1-cycle break device, the address on the DATA ADDRESS lines is provided by a register in the peripheral and represents the memory address of the data to be transferred. The address on the DATA ADDRESS lines is sampled by the TP4 pulse. The OMNIBUS CPMA DISABLE line is asserted by the data break interface at TP4 to enable the DATA ADDRESS information to be placed on the MA lines. 1 = GND.
BRK RQST*	This signal is asserted (brought to ground) by the peripheral when it is ready for a word transfer. When BRK RQST is present at INT STROBE time, the data break operation is entered. The OMNIBUS INT IN PROG line is asserted, and a load enable signal is provided for the data break interface break memory address (BKMA) register.

*Pertains to Data Break interface only.

SIGNAL NAME	DESCRIPTION
ADD ACCEPTED*	This signal is generated by the data break interface when a BRK RQST signal has initiated the data break operation. ADD ACCEPTED is used in the peripheral to clear the BRK RQST flip-flop. Pulse = GND.
MB INCREMENT*	When this signal is at ground level during the true break cycle, the contents of the memory location are acted upon as outlined in the following table.

MB INCREMENT	DATA OUT	Operation Performed	Descriptive Term Used for Operation
Low	Low	Contents of the memory location are incremented.	MB Inc
Low	High	Data on the DATA 00-11 lines is added to the contents of the memory location.	Add to Memory (ADM)

CA INCREMENT INH*	When this signal is asserted (grounded) during the CA cycle of a 3-cycle data break, the CA is not incremented.
3-CYCLE*	This signal is transferred from the peripheral to the data break interface to notify the interface logic to set either the WC flip-flop (3-cycle transfer) (ground input) or the B flip-flop (1-cycle transfer).
WC OVERFLOW*	The interface transfers this signal to the peripheral to notify it that the word count location in memory has become zero and that the data transfer should end. The signal is also present when overflow occurs during MB increment or ADM. Pulse = GND.
EXT DATA ADD 0-2*	These three lines are used when a KM8-E Memory Extension and Time Share interface is included in the basic PDP-8/E. The peripheral uses the lines to indicate the particular memory field involved in the transfer. During a 3-cycle data break, WC and CA cycles always occur in field 0, while only the B cycle occurs in the field specified by the extended data address. 1 = GND.
B INITIALIZE 2*	This positive signal clears all flags in the peripheral and is essentially the INITIALIZE signal of the OMNIBUS. It is used by the break device in lieu of B INITIALIZE 1 so as to reduce loading on the latter.

* Pertains to Data Break interface only.

APPLICATION

The nature of the external bus and its relationship to the OMNIBUS have been presented. Now, the use of the bus must be fully explored. First of all, the user wants to transfer data between his peripheral and the computer's memory. He can do this in any one of three ways—programmed I/O transfers, program interrupt transfers, or data break transfers. The basic ideas behind all three methods of data transfer have been discussed in previous chapters, and the user should be familiar with these before proceeding any further in this chapter.

Programmed I/O Transfers

Figure 10-3 is a logic block diagram that shows the more important signals involved in a programmed I/O transfer. The transfer process is, of course, similar to that which takes place between the computer and a peripheral interfaced to the OMNIBUS. Each peripheral has a flag flip-flop that is set when the peripheral is ready to receive or send information. A programmed IOT instruction is used to check this flip-flop. The program enters a waiting loop until the peripheral is ready. When the flag flip-flop is set, IOT XXA (illustrated on the block diagram) asserts the SKIP bus. The program then skips to an instruction that transfers program control to a servicing subroutine. The subroutine carries on the IOT dialogue between peripheral and processor. Although the general process is similar, both the method of peripheral selection and the use of the SKIP function differ for external bus peripherals.

The method of peripheral selection will be examined first. As shown on the block diagram, the peripheral contains a device selector and an IOT generator. The device selector monitors the BMB03 through BMB08 lines. These lines are merely the buffered OMNIBUS MD03 through MD08 lines. Thus, when an IOT instruction is issued, BMB03 through BMB08 carry the code for a particular peripheral. The device selector responds to the code by producing a DEVICE SELECTED signal. This signal is applied to the IOT generator. Unlike the OMNIBUS peripherals, external bus peripherals require BIOP pulses to carry out the operations specified by the IOT instructions. These pulses are generated in the Positive I/O Bus interface and reflect the information present on the OMNIBUS MD09, 10, and 11 lines. BIOP pulses are applied to the IOT generator where they are regenerated as IOT pulses that initiate operations within the peripheral. Figure 10-3 shows that the BIOP pulses (1, 2, and 4) are generated only when the following conditions are met: the OMNIBUS I/O PAUSE line is asserted, indicating that an I/O transfer is to take place; the INTERNAL I/O line is negated, indicating that the I/O transfer is to or from an external bus peripheral; the MD09, or MD10, or MD11 line is asserted. Thus, if the first two conditions are met, and MD09 is asserted, BIOP4 is generated. Similarly, BIOP1 is generated when MD11 is asserted. Each BIOP pulse is regenerated as an IOT pulse within the peripheral. Thus, BIOP1 becomes IOT XX1; BIOP4 becomes IOTXX4. (In the preceding notation, XX represents the particular device selection code).

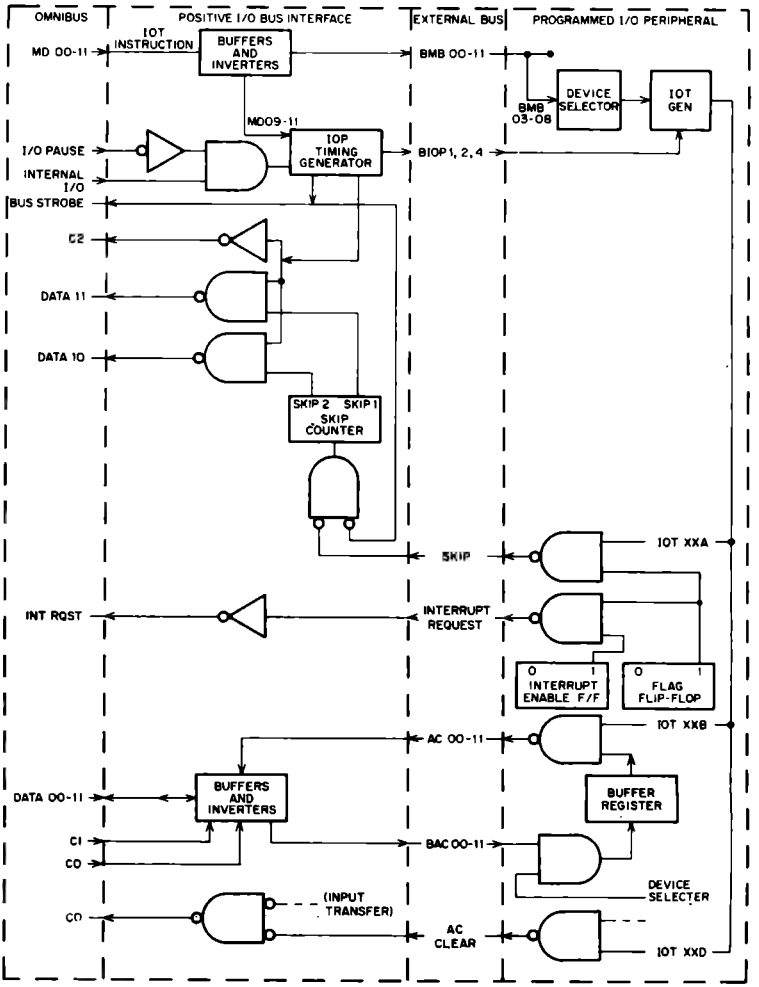


Figure 10-3 Block Diagram, Programmed Data Transfer

Specific IOT pulses perform specific operations in the peripheral. The primary use of specific IOT pulses, as well as the relationship between IOT, BIOP, and MD bits, is as follows:

MD Bit	BIOP	IOT	Primary Use
09	BIOP4	IOT XX4	Reading, loading, and clearing buffers.
10	BIOP2	IOT XX2	Clearing flags; clearing AC.
11	BIOP1	IOT XX1	Sampling flags; skipping.

These relationships are standard within DEC and are certainly not mandatory for the user. He may decide to use IOT XX1 to clear a flag, or he might wish to use IOT XX4 to sample a flag. He may even use combinations such as IOT XX5 by generating both BIOP1 and BIOP4 with the same IOT instruction. In any event, it is wise to develop some standard such as that expressed above.

The discussion so far has shown how the peripheral is selected and how operations are initiated by the IOT instruction. The use of the SKIP function must now be explained. One should recall that a flag flip-flop in the peripheral is sampled by an IOT pulse and a skip is effected. IOT XXA checks the status of this flip-flop (XXA is used here, rather than XX1, to emphasize the fact that "A" may be 1, 2, 5, or whatever the user wishes). When the peripheral is ready, the SKIP bus is asserted. A strobe signal from the IOP timing generator clocks the skip counter, a two-stage binary counter. This strobe signal is generated near the end of the BIOP pulse. If one BIOP pulse is generated by the IOT instruction, the skip counter is clocked only once and the SKIP 1 line is asserted. A control signal, produced at the end of the BIOP pulse, then asserts both the C2 line and the DATA 11 line of the OMNIBUS. When the C2 line is asserted, with C1 negated, DATA + PC goes to PC. In other words, the PC is incremented by one and the program skips one instruction. Figure 10-4 is a timing diagram of the SKIP function and is helpful in visualizing the process. Note that two strobes are generated, each performing the function shown on the diagram.

It should also be noted on the timing diagram that the CP operation is halted from TP3 to the beginning of the second BUS STROBE. This BUS STROBE, which occurs after NOT LAST TRANSFER has been negated, generates INT STROBE, which restarts the CP. This is the disadvantage inherent in external bus interfacing—the CP must remain inactive while BIOP pulses are generated and control lines are activated. Thus, while OMNIBUS I/O transfers are accomplished in 1.2 microseconds, the minimum time required for external bus I/O transfers is 2.6 microseconds when only one BIOP pulse is generated by the IOT instruction. Suppose two BIOP pulses are generated. The user might want to skip two instructions in the program, for example. In this case, the IOP timing generator produces three BUS STROBES. Each of the first two BUS STROBES clocks the skip counter. Thus, the SKIP 2 line is asserted. Again, C2 is asserted by a control signal, but now DATA 10 rather than DATA 11 is brought low. BUS STROBE three then enables DATA + PC to the PC, and the program skips the next two instructions. This may be advantageous for a specific application, but note the increased transfer processing time—it is now 3.6 microseconds. If three BIOP pulses are used, the CP is halted for 4.6 microseconds. It must be remembered that these are minimum values. Figure 10-4 shows that the time from

TP3 to the beginning of BIOP1 is variable, as is the width and the pulse separation when more than one BIOP is issued by the IOT instruction.

The user may want to increase IOP width and/or separation for specific applications. He should bear in mind, however, that this affects processing time and may slow the computer appreciably. The width and separation are controlled by separate potentiometers on the interface.

The minimum allowable values are 800 nanoseconds for pulse width and 200 nanoseconds for pulse separation. The resistor values in the timing circuits allow the user to increase these values to five times the minimum.

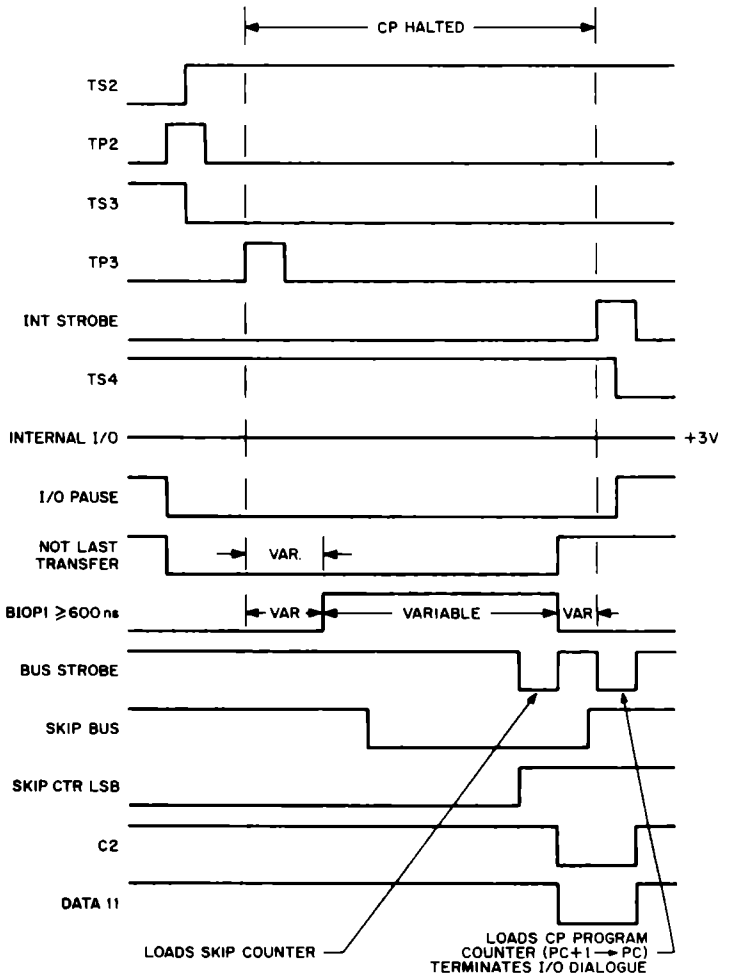


Figure 10-4 Timing Diagram, Skip Bus Application

Figure 10-5 shows another timing diagram, this one illustrating a data word transfer. Conventionally, BIOP4 is being used to accomplish the transfer. Note that the BIOP pulses do not occupy specific time slots. Thus, if BIOP1 and BIOP2 are not required, as in this example, the IOP timing generator produces BIOP4 without any delay. Again two BUS STROBES are generated, each performing the indicated function. The C1 control line must be asserted to indicate an input data transfer. This is done if data is placed on any external bus AC line. The data is buffered and inverted, and placed on the OMNIBUS DATA 0-11 lines (one of these lines, DATA XX, is illustrated). With only C1 asserted, a 1's transfer to the AC is carried out. If a jam transfer to the AC is desired, the C0 control line must be asserted along with C1. This can be done only if the external bus AC CLEAR line is brought to ground, which can be accomplished either by gating IOT4 in the peripheral or by other means that the user might select. Do not permanently ground AC CLEAR.

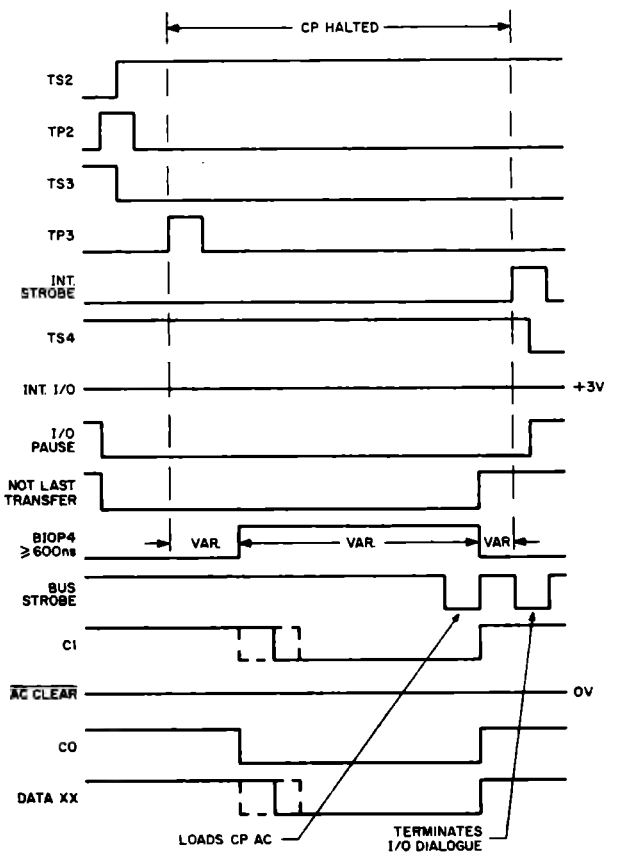


Figure 10-5 Timing Diagram, Programmed Data Transfer

Program Interrupt Transfers

Program interrupt devices are connected to the external bus exactly as are programmed data transfer devices, and use the very same signals. The entire program interrupt process proceeds just as it does for OMNIBUS peripherals. The peripheral requests an interrupt by setting its flag flip-flop. The processor honors the request and services the peripheral in a program subroutine. The data transfer is accomplished by programmed transfers, as explained in the preceding section. The user must provide means of asserting (grounding) the external bus INTERRUPT REQUEST line when the peripheral is ready for a transfer. Figure 10-3 shows the peripherals flag flip-flop controlling the INTERRUPT REQUEST line. The SKIP line must also be used when more than one such interrupt device is connected onto the external bus and is utilized as detailed in the preceding section.

Data Break Transfers

Data break transfers involving peripherals on the external bus are accomplished in much the same way as transfers involving OMNIBUS peripherals. Refer to figure 10-6, which is a logic block diagram that shows the more important functional blocks and signals involved in a data break transfer. Figure 10-7 shows the timing relationship of the major signals and, along with the block diagram, should be referred to throughout this discussion.

An important feature that is illustrated on the block diagram is the circular flow of data between OMNIBUS and peripheral. The peripheral receives data from the computer by way of the OMNIBUS MD lines and the external bus BMB lines. It sends data to the computer by way of the external bus DATA 00-11 lines and the OMNIBUS DATA 0-11 lines. Thus, there is a need for both interface boards when data break peripherals are used. In addition to providing a data path, the Positive I/O Bus interface provides BIOP pulses. These pulses, as in programmed data transfers, are used in conjunction with a device selector to produce IOT pulses in the peripheral. However, data break IOT pulses are used only to initiate the data break operation and, once the operation has started, program control of the peripheral ceases. Therefore, the program provides IOT instructions that tell the peripheral, via the IOT pulses, to perform any preliminary operations that may be required. When the peripheral has followed these instructions and is ready to either send or receive data, it requests a Data Break. The data break interface logic then assumes control over the operation.

When the Data Break interface receives the BRK RQST signal from the peripheral, it uses the next INT STROBE from the OMNIBUS to assert the OMNIBUS BRK IN PROG line. At TS4 the interface checks the priority network to make sure no higher priority device is present in the system. If none is present, the interface asserts two OMNIBUS processor-control lines at TP4. Assertion of these lines—CPMA DISABLE and MS, IR DISABLE—causes the processor to load its CPMA register and suspend operation, while the peripheral interface BKMA register assumes control of the OMNIBUS MA lines (as with OMNIBUS data break peripherals, the break can occur at the end of any processor major state). At the same time—TP4—the interface major state control logic enters either

the word count cycle or the break cycle, depending on the state of the 3 CYCLE signal from the peripheral. If this peripheral is a single-cycle break device, the interface logic enters the B (break) cycle. The B BREAK signal is generated and sent to the peripheral where it either loads the output buffer register with data or places data from this register on the lines to the interface logic. If the transfer direction is from the peripheral to memory, data from the buffer register is placed on the DATA 00-11 lines. At TS2 this data is placed onto the OMNIBUS DATA 0-11 lines. The next TP2 loads the data into the MB, providing the BREAK DATA CONT has not been asserted, and the MB contents are then written in memory.

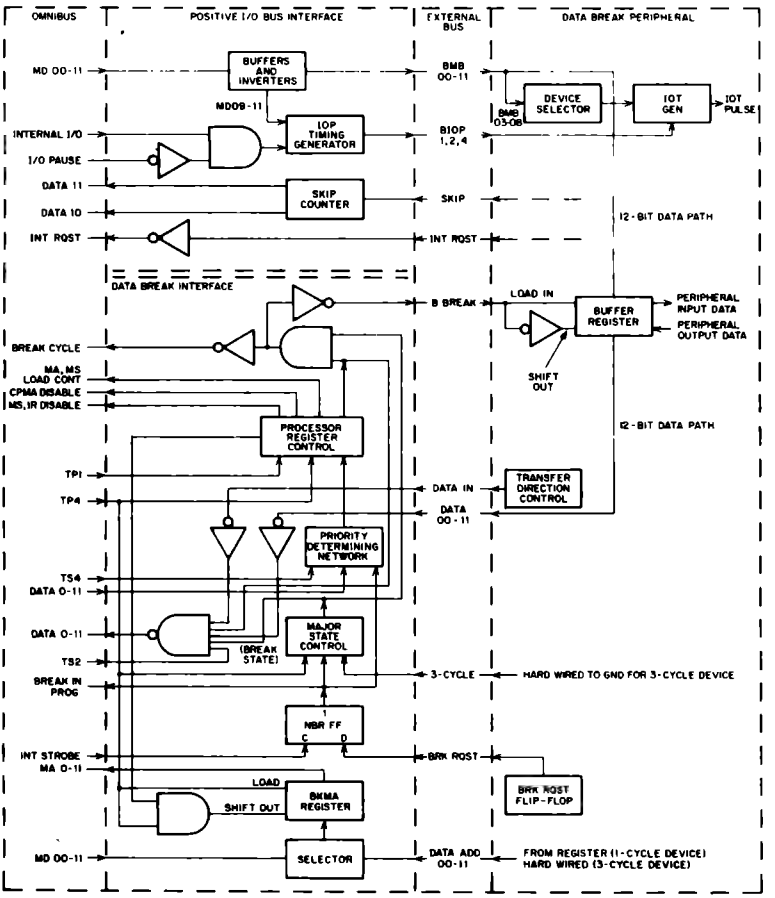


Figure 10-6 Block Diagram, Data Break Transfer

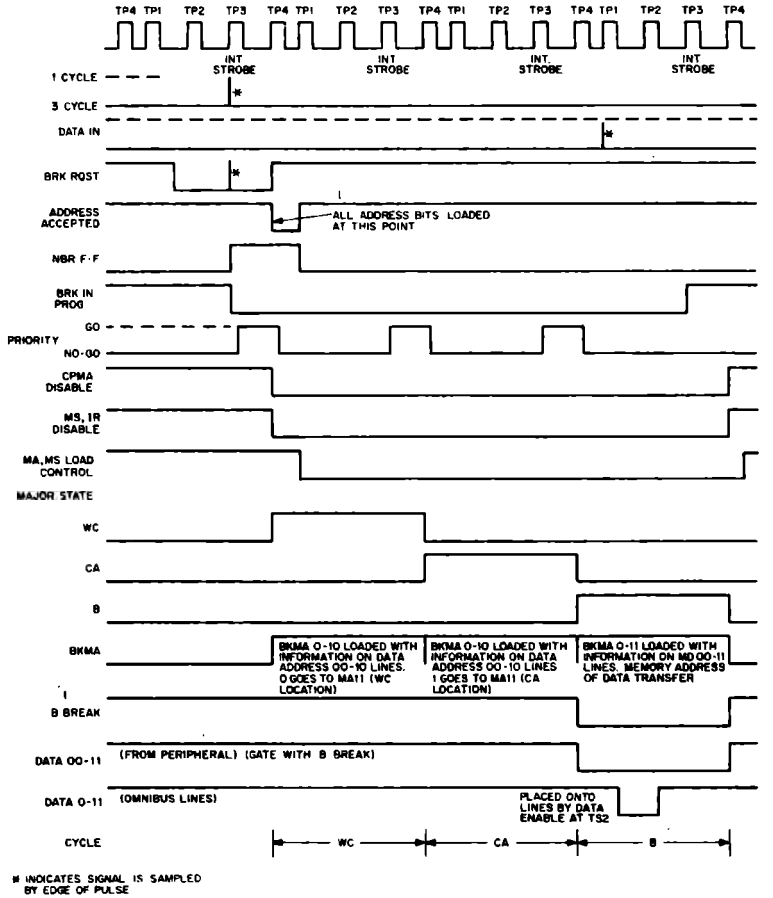


Figure 10-7 Timing Diagram, Data Break Transfer

Figure 10-7 indicates a peripheral-to-computer data transfer. It also shows the timing for a 3-cycle, rather than a single-cycle, break device, and indicates that the WC and CA cycles must be completed before B BREAK is generated at the beginning of the B cycle. In addition, the timing illustrates a very important and advantageous feature of the interface logic—priority is checked not only before entering the WC cycle, but also before entering the CA and B cycles. This means that high priority devices can override lower priority devices even in the middle of a break operation. As many as 12 Data Break peripheral interface boards can be inserted into the OMNIBUS. Each peripheral must first be assigned a priority level. Each interface contains an identical priority network that is wired in a unique fashion, so as to reflect the priority

level assigned to the peripheral. During any TS4 of a data break cycle, the highest priority interface, of those interfaces whose peripherals have made break requests, receives a 'go' signal from its priority network. All other interfaces whose peripherals have made requests receive a 'no-go' signal, and must wait until the next TS4 for another priority check.

This unique priority system allows the user to utilize his data transfer system in a most efficient manner. However, he must pay careful attention to the manner in which he assigns priorities. An essential rule should be: assign priorities in relation to the length of time data is available at the peripheral. For example: peripheral A and peripheral B both assert their BRK RQST lines at some time between TP1 and TP3 of a particular processor cycle. Peripheral A has been assigned a higher priority than peripheral B; and, thus, takes control of the break operation. If 'A' is a 3-cycle device, assuming no other higher priority devices are present, it does not relinquish control until it has transferred the data word during the B-cycle. During TS4 of this cycle, 'B' receives a priority 'go' signal and takes control of the operation at TP4. If 'B' is also a 3-cycle break device it must now go through WC and CA before the actual transfer can begin.

Seven to eight microseconds has elapsed from the moment that 'B' requested a break until the moment when its interface generates B BREAK. If 'B' is a very high speed device, the data that was present in its output register at the time of the break request may no longer be present when B BREAK is finally generated. The solution in this example is obvious (if 'A' is a slower device than 'B', that is): assign 'B' a higher priority than 'A'. OMNIBUS data break peripherals are also equipped with priority networks, although these generally differ from the network of the Data Break interface. Nevertheless, all types work together, and the only limitation is on the total number of priority networks, 12.

While OMNIBUS data break peripherals are generally single-cycle devices, those data break peripherals interfaced to the external bus are, for the most part, 3-cycle devices. Because the WC and CA registers of a 3-cycle device are located within core memory, the Data Break interface BKMA register is used differently from its counterpart in an OMNIBUS peripheral. Figure 10-7 indicates the use of the BKMA register during each cycle of the 3-cycle break operation. During WC, the hard-wired memory address of the word count register is loaded into the BKMA. The only restrictions on this address are that bit 11 be a '0' and that the memory location be in memory field 0.

The word count in memory is brought out, incremented, and deposited back in memory. During CA, the same hard-wired address is loaded into BKMA 0-10, and BKMA11 is asserted. The result is the memory address of the current address register. The current address is brought out of memory, incremented, and placed on the MD lines. Not only is this address deposited back in the CA register, but it is also loaded into the BKMA register at the beginning of the B cycle. Thus, the current address specifies the memory address to which, or from which, the data is to be transferred.

In the preceding explanation, data has been presented as a 12-bit data word that is transferred unaltered to or from a specified memory location. However, the interface logic provides the user with two features that somewhat alter this idea. The first, MB INCREMENT, allows the user to transfer a single bit of information via the OMNIBUS DATA 11 line. The result is an incrementation of the data contained in the specified memory location. The peripheral must cause the external bus MB INCREMENT line to be asserted with the DATA OUT line asserted. Each time MB INCREMENT is asserted, the interface logic asserts the OMNIBUS DATA 11 line and the data in the memory location is incremented.

The second feature, Add to Memory (ADM), allows the user to alter the 12-bit data word in memory. Specifically, the incoming data word and the data contained in the addressed memory location are added in the processor memory buffer. The result is then returned to the addressed location. As with MB INCREMENT, the peripheral must cause the MB INCREMENT line to be asserted; but now the DATA OUT line must be negated. The result is ADM. Table 10-1 lists the possible states of the DATA IN line and the MB INCREMENT line, and the resulting data transfer. L indicates that the line is asserted (grounded), while H indicates that the line is negated.

Table 10-1. Data Control Lines (Data Break Interface)

DATA OUT	MB INCREMENT	TYPE AND DISPOSITION OF DATA
L	H	12-bit data word transferred from the addressed memory location to the peripheral buffer via the BMB lines.
H	H	12-bit data word transferred from the peripheral buffer to the addressed memory location via the DATA00-11 lines.
L	L	MB INCREMENT. The contents of the addressed memory location are incremented.
H	L	Add to Memory (ADM). The twelve-bit data word on the DATA 00-11 lines is added to the contents of the addressed memory location, and the result stored in the addressed memory location.

INTERFACING TECHNIQUES

At this point the user should have a good understanding of what the external bus is and the way that he wants to use it. The remainder of this chapter is devoted to the technique of interfacing the bus to the user's peripheral.

When the user purchases a PDP-8/E, he selects the options he requires. Assume that he wants both interfaces. Digital Equipment Corporation then supplies the boards, the interconnecting cables, and the connectors on either end of each cable. The customer specifies whether the cable should be shielded flat cable or round or flat coaxial cable. In addition, he selects the cable length from a variety of standard lengths made available by DEC. The connectors on either end of the cable are standard DEC connectors and those on the free end will connect into any peripheral control manufactured by DEC.

PDP-8/I and 8L-type Peripherals

If the user has a PDP-8/I or PDP-8/L-type peripheral control, the PDP-8/E can be easily interfaced to it. If the control is a data break peripheral control, the user first plugs the two interface boards (with the cables already connected to the board) into the OMNIBUS, in any available slots. He then inserts the peripheral control connectors into the appropriate slots in the peripheral control connector block. In general, the appropriate slot is determined as follows:

Cable Connector		Peripheral Module Slot	Signals
I/O Bus Interface	Data Break Interface		
Cable 1		A1	BAC
Cable 2		A2	BMB
Cable 3		A3	AC INPUT
	Cable 4	A4	DATA ADDRESS
	Cable 5	A5	DATA BITS

(The above information is generally true; however, because peripheral control module arrangements sometimes vary, the user should check the peripheral control interconnection information to see that the stated correspondence is correct.)

If the peripheral is a programmed I/O device, only the Positive I/O Bus interface must be inserted into the OMNIBUS. In this instance, cables 1, 2, and 3 of the board connect as indicated above. If several peripherals are being connected to the bus, they are connected as indicated in figure 10-1. Table 10-2 lists the external bus signals and provides the connector pin assignment for each signal line. The location of a particular pin can be determined as follows: if the connector board is held so that the connector pins are to the left and the keying cutouts appear as shown in Figure 10-8 (the longer cutout on the bottom), pin A is the topmost pin, pin V is the bottommost, and side 2 is the nearer side. Pins are lettered A through V (excluding G, I, O, and Q). This convention of pin identification also applies to the M-series and G-series connectors and modules.

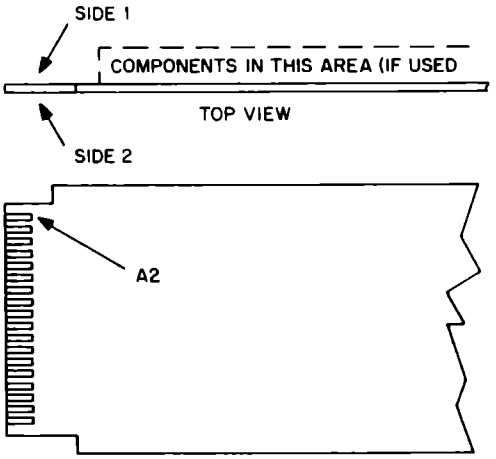


Figure 10-8 Connector Pin Identification

Table 10-2 External Bus Connector Pin Assignments

POSITIVE I/O BUS INTERFACE

Signal Name			Connector Pin
Cable 1	Cable 2	Cable 3	
BAC 00	BMB00(1)	AC 00*	B1
BAC 01	BMB01(1)	AC 01*	D1
BAC 02	BMB02(1)	AC 02*	E1
BAC 03	BMB03(0)	AC 03*	H1
BAC 04	BMB03(1)	AC 04*	J1
BAC 05	BMB04(0)	AC 05*	L1
BAC 06	BMB04(1)	AC 06*	M1
BAC 07	BMB05(0)	AC 07*	P1
BAC 08	BMB05(1)	AC 08*	S1
BAC 09	BMB06(0)	AC 09 *	D2
BAC 10	BMB06(1)	AC 10*	E2
BAC 11	BMB07(0)	AC 11*	H2
BIOP1	BMB07(1)	SKIP BUS*	K2
BIOP2	BMB08(0)	INT RQST BUS*	M2
BIOP4	BMB08(1)	AC CLEAR BUS*	P2
BTS3	BMB09(1)	B RUN (0)	S2
BTS1	BMB10(1)		T2
B INITIALIZE 1	BMB11(1)		V2

DATA BREAK INTERFACE

Signal Name		Connector Pin
Cable 1	Cable 2	
DATA ADD 00*	DATA 00*	B1
DATA ADD 01*	DATA 01*	D1
DATA ADD 02*	DATA 02*	E1
DATA ADD 03*	DATA 03*	H1
DATA ADD 04*	DATA 04*	J1
DATA ADD 05*	DATA 05*	L1
DATA ADD 06*	DATA 06*	M1
DATA ADD 07*	DATA 07*	P1
DATA ADD 08*	DATA 08*	S1
DATA ADD 09*	DATA 09*	D2
DATA ADD 10*	DATA 10*	E2
DATA ADD 11*	DATA 11*	H2
BRK RQST*	3 CYCLE*	K2
DATA OUT*	CA INCREMENT INH*	M2
B BREAK (0)	BWC OVERFLOW	P2
ADD ACCEPTED	EXT DATA ADD 02*	S2
MB INCREMENT*	EXT DATA ADD 01*	T2
B INITIALIZE 2	EXT DATA ADD 00*	V2

Signals marked * are input signals, and are asserted at ground. Unmarked signals are output signals, and are asserted at +3V.

Note: Be sure to ground all other pins except U1, V1, A2, and B2.

Customer Peripherals

The interfacing technique becomes more complicated when the user desires to connect his own peripheral. The user must decide if he wants to use the cables and connectors that are supplied with the interface board. If he elects to do so, he must use DEC connector blocks that accommodate the BC08J interface cables. He must then provide a mechanical interface between the DEC connector block and the peripheral input connectors, and, in addition, must provide an electrical interface between the external bus and the peripheral. The designer of the interface has to consider questions of voltage levels, loading criteria, mechanical compatibility, etc. The task of interfacing is greatly simplified if the customer makes use of DEC's line of M-series modules and compatible H-series connector blocks and mounting panels.

DEC Logic Module Interfacing

As an example of the application of DEC modules to the user's interfacing problem, consider the following: the peripheral, whether a data break device or a programmed transfer device, must be able to recognize its device code and respond in some way to that code. The device code is contained in the BMB03-08 bits, which are provided by cable 2 from the Positive I/O Bus interface. The user can design his own integrated circuit or discrete component interface to decode the BMB03-08 bits and provide a Device Selected signal. He can then breadboard the circuit, test it, build it, and install it in some kind of connector,

which he then has to connect to the interface cable. Alternately, he could use a DEC M103 Device Selector module, which, in addition to providing the correct Device Selected signal, accomplishes most of the interfacing, provides diode clamp protection, and even provides extra logic gates for the customers use.

The M103 module is shown in Figure 10-9. Assume that the user assigns device code 14 to his peripheral. The selected BMB lines provide an enabling signal for gate 2 whenever device 14 is selected. The module also provides the peripheral with the necessary IOT pulses, as is indicated in Figure 10-9. The output signals are available for immediate use.

The M-series modules use TTL logic, and the input loading/output drive requirements are given in number of unit loads. A unit load is defined as follows: in the logic 0 state, the driver must be able to sink 1.6 milliamps (maximum) from the unit load's input circuit, while maintaining an output voltage equal to or less than 0.4 volts; in the logic 1 state, the driver must maintain an output voltage equal to or greater than 2.4 volts, while supplying the unit load with a leakage current of no more than 40 microamps. The M103 is capable of driving 37 TTL unit loads at the IOT outputs, while the DEVICE SELECTED output can drive 16 TTL unit loads.

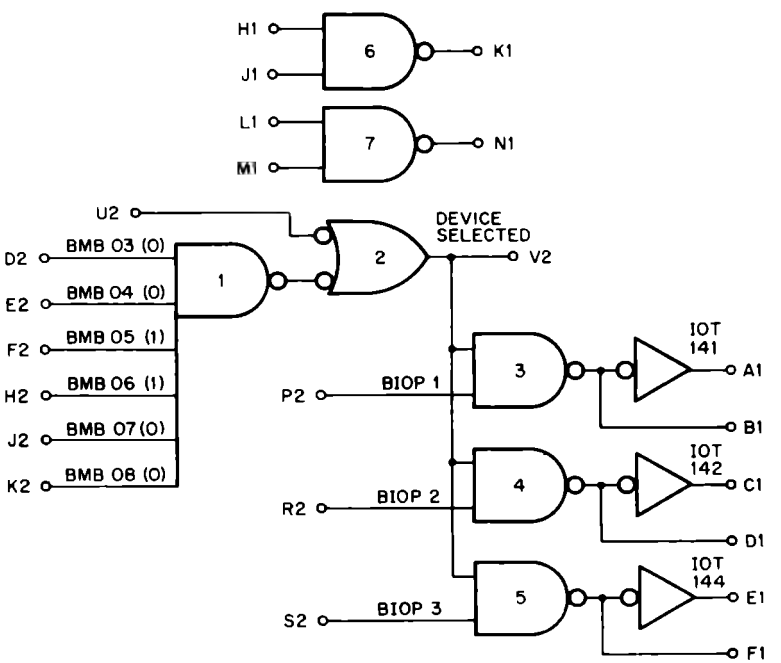


Figure 10-9 M103 Device Selector Module

Besides getting the peripheral selected, it is also necessary to transfer data. If, for example, data is to be sent to the PDP-8/E accumulator, it must be fed onto the external bus AC00-11 lines. The M624 Bus Driver module can provide this capability. This module, shown in Figure 10-10, contains 15 bus drivers. Twelve have a common gate line and can be used as shown in Figure 10-11. An output register in the peripheral is loaded with the data that must be transferred. The IOT pulse can then clock the data onto the AC lines. The user provides the proper input drive for the bus drivers. The M624 presents the following input TTL unit loads; 12 loads at the clock input (IOT 14X line); 1 load at the data inputs.

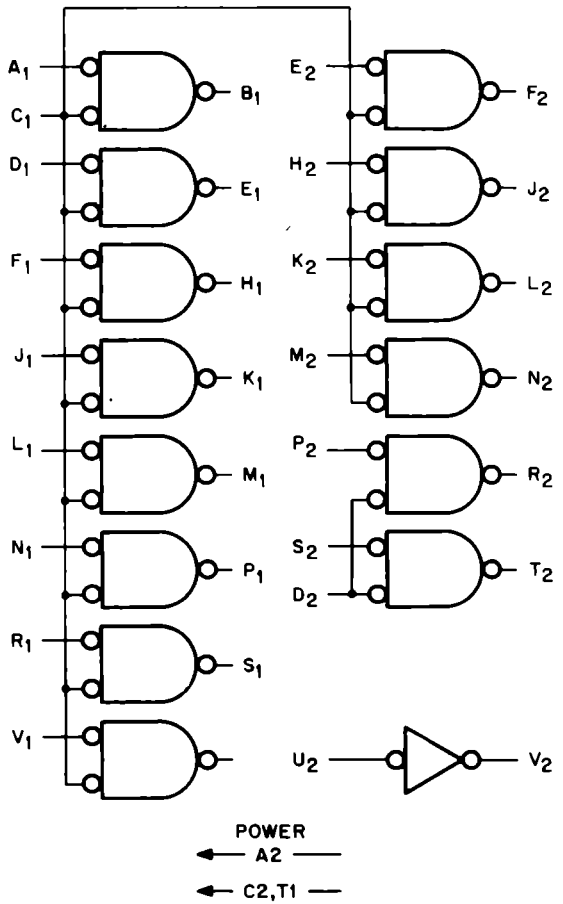


Figure 10-10 M624 Bus Driver Module

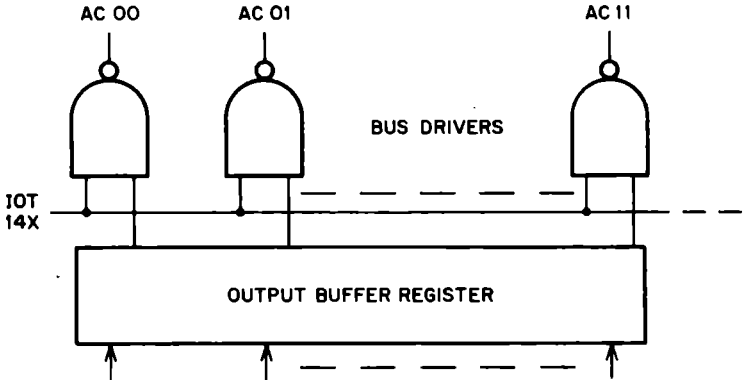


Figure 10-11 Bus Driver Application

The user also has to transfer data from the accumulator. Therefore, he must get the data on the BAC 00-11 lines into the peripheral buffer register. While doing this, he must take care that he does not place an excessive load on any BAC line. If he uses an M101 Bus Data interface, shown in figure 10-12, he can be sure of maintaining proper loading of these lines. Figure 10-13 shows how this interface must be used. Twelve of the gates are connected to the 'device select' line. When the device (XX) is selected, the BAC 00-11 data is applied to the input lines of the buffer register. IOT XXY then loads the register with the data word. The M101 presents the following input TTL unit loads: 15 loads at the clock input (device select line); 1 load at the data inputs.

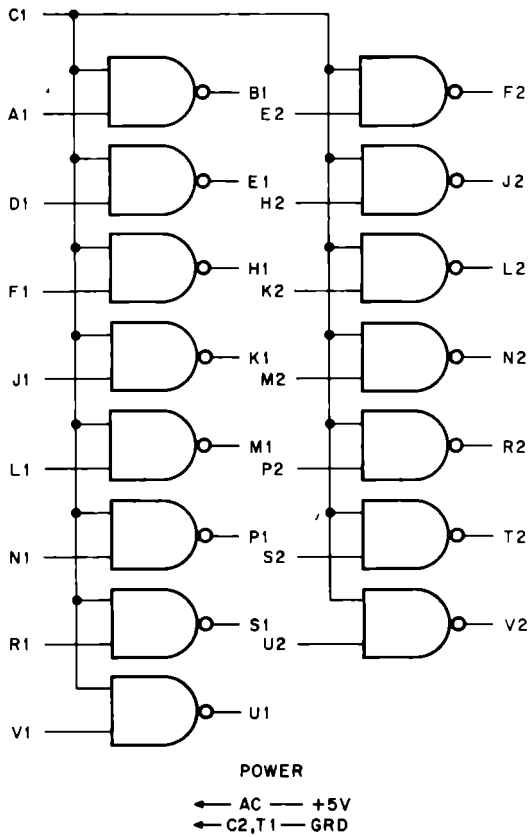


Figure 10-12 M101 Bus Data Interface

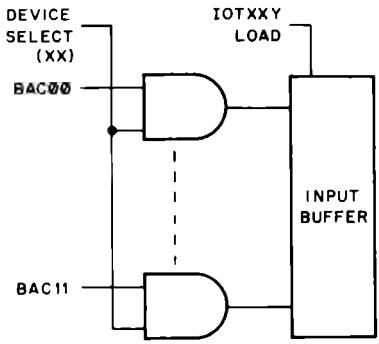


Figure 10-13 Bus Data Application

A variety of M-series modules are available from which the user can select to meet his needs. A list of these modules is given at the end of this chapter. This list should prove helpful in the design of interfaces. Attention is also directed to DEC's DIGITAL LOGIC HANDBOOK available from all sales offices.

M-series modules greatly simplify the task of electrical interfacing. However, this is only part of the problem. The external bus and the peripheral must also be mechanically interfaced. DEC's H-series of connector blocks and mounting panels are extremely valuable in this application. The following example takes the user through a mechanical interfacing procedure to acquaint him with the technique.

The user wants to interface a programmed transfer type of peripheral to the external bus. He wants to use the cables provided with the Positive I/O Bus interface, and he intends to solve part of the interfacing problem by using M-series modules. Not only does he need a connector block into which he can insert either an M-series module, or the connector on cables 1, 2, and 3, but he also needs something on which to mount each connector block. DEC's H803 connector block, providing slots for eight modules, accepts the connectors, the M-series modules, and even DEC's A-, K-, and W-series of modules. To mount these connector blocks, H911 mounting panels can be used. This mounting panel contains eight H803 connector blocks and can be mounted in a standard 19-in. equipment rack. Figure 10-14 shows an H911 mounting panel containing two H803 connector blocks. The wire-wrap pins of the connector blocks face the front of the mounting panel. Cable 1 from the Positive I/O Bus interface is pictured as connecting into slot A1. This is in accordance with the convention presented previously. Cable 2 and cable 3 (omitted for clarity) would be inserted into slots A2 and A3, respectively.

If a data break peripheral were being used, cable 1 from the KD8-E interface would be inserted into slot A4, while data break cable 2 would be inserted into slot A5. The M-series module is shown as being inserted into slot A6. It may be inserted into any available slot, except B1 through B5. Finally, the peripheral I/O cable is shown as being inserted into slot A7 (again just an arbitrary assignment). The peripheral I/O cable must be equipped with a connector that is compatible with the H803 connector block. Either an M903 connector (for use with shielded Mylar) or an M904 connector (for use with coaxial cable) is recommended.

Slots B1, B2, and B3 should be wired in parallel with A1, A2, and A3 so that the I/O bus can be continued to additional peripheral controls. In general, module layout is primarily a matter of common sense. The convention just given is a standard with DEC and may or may not meet with the user's approval. Customers should, nevertheless, always attempt to make parallel connections of the Positive I/O bus interface cables to facilitate possible future expansion.

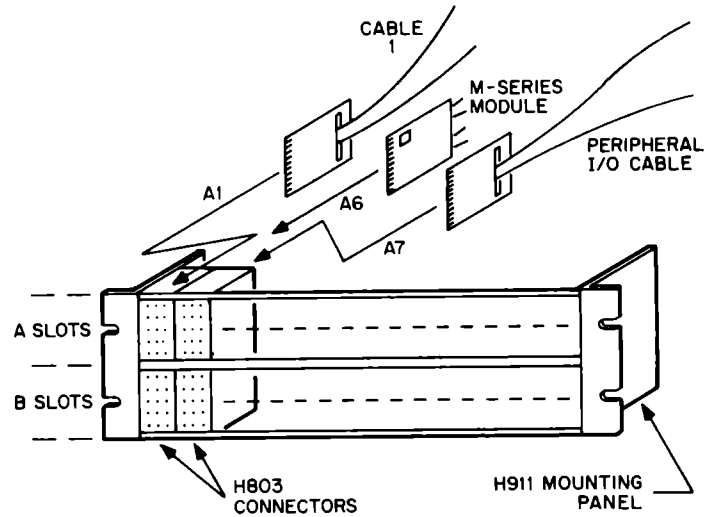


Figure 10-14 Interface Hardware

Connections between the various cable connectors and the M-series modules are made by selective wiring of the H803 connector blocks. The following suggestions and requirements are provided to help reduce the mounting panel wiring time. The connectors should be wired in the order given in steps 1 through 4.

1. The H803 mounting blocks have wire-wrap pins, thereby eliminating the problems associated with soldering, while at the same time providing highly reliable, long lasting connections. DEC recommends #30 AWG, Teflon-coated solid wire for connector block wiring. Smaller wire may be used if many connections are to be made to a single lug. A wire-wrapping tool must be used to wire the connector pins. DEC type H810 pistol grip hand wire-wrapping tool is designed for wrapping #24 or #30 solid wire on Digital-type connector pins (check the DIGITAL LOGIC HANDBOOK for further information about this tool and any other hardware mentioned in this chapter).

CAUTION

Whenever a wire-wrapping tool is used on a mounting panel containing modules, steps must be taken to avoid voltage transients that can burn out components. A battery-powered or air-operated tool is preferred, but even with these tools static charge can build up and burn out semiconductors. If the modules remain in the connector panel during wiring, ensure that the wire wrap tool is electrically grounded. Whenever soldering is done on a mounting panel containing modules, a 6V soldering iron should be used.

2. Certain connector pins on cable connectors and modules are reserved for specific functions. Cable connector pins are reserved as follows.

Signals:	B1, D1, E1, H1, J1, L1, M1, P1, S1, D2, E2, H2, K2, M2, P2, S2, T2, V2.
Grounds:	A1, C1, F1, K1, N1, R1, T1, C2, F2, J2, L2, N2, R2, U2.
Not Used:	U1, V1, A2, B2.

Module pins are reserved as follows:

Positive dc voltage:	A2 (usually +5V)
Negative dc voltage:	B2 (usually −15V)
Ground:	C2, T1.

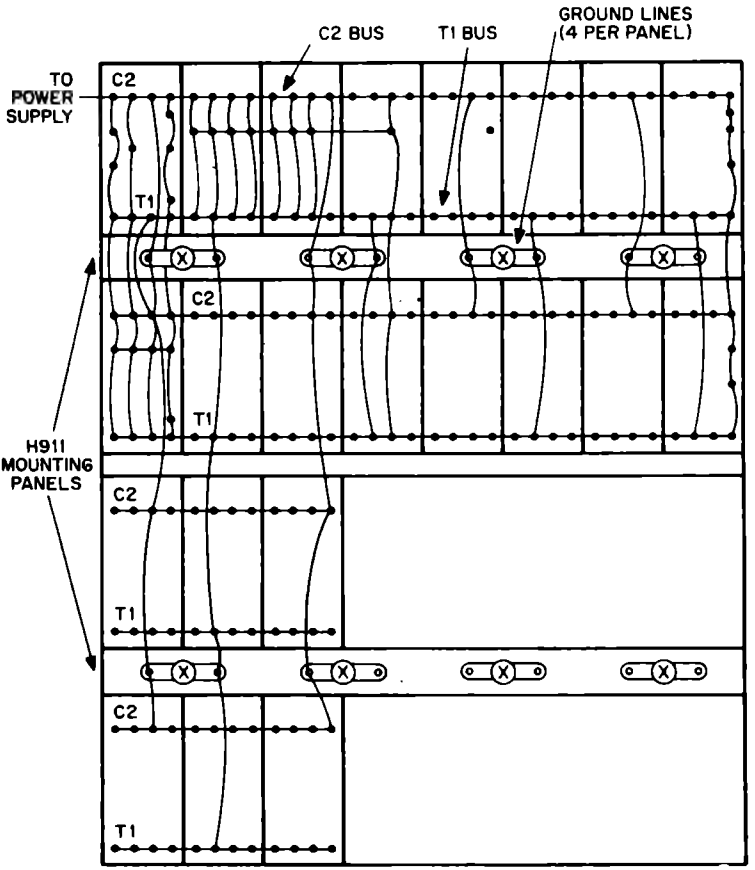
Some form of bus strip (such as DEC 933 Horizontal Bussing Strips) should be used to make all connector power connections and all horizontally bussed signal connections. Negative DC voltage should be wired to pin B2 of module connectors only if the voltage is required by the module.

3. Adequate grounding must be provided. The user should not be concerned with the question of ground loops. At the frequencies dealt with in digital logic, many parallel paths are of utmost importance. There must be ground continuity between cabinets, and between the logic assembly and any equipment with which the logic connects. Continuity between mounting panels and between ground pins on the various connector blocks is achieved when the following instructions have been carried out. These instructions are illustrated in Figure 10-15.
   a. Vertical grounding wires must interconnect each chassis ground lug with pin C2 and pin T1 grounds. Start these wires at the uppermost mounting panel and continue to the bottom panel. Begin by connecting C2 pins, then T1 pins, alternating thereafter. Space the wires about two inches apart, so that each of the chassis-ground lugs is in line with one of the wires. Each vertical wire should make three connections at each mounting panel.
   b. Connect pin C2 of each module to T1 of the same module, making connections to all other pins to be grounded along the way. Connect T1 of each module to C2 of the module beneath. Ensure that connections are made to the ground pins on the signal connectors.
   c. Bus ground pins horizontally wherever possible.

4. After ground connections have been made, connect all signal wires in any convenient order. Point-to-point wiring produces short wire lengths, installs quickly, is easy to trace and change, and generally results in better appearance and performance than cabled wiring.

   Certain restrictions and criteria must be observed when interfacing to the external bus. These are encountered when interfacing both with DEC modules and connector blocks, and with cus-

tomer-designed circuits. These requirements are contained in the section dealing with restrictions and criteria. Users are strongly advised to consult this section prior to completing their interface package.



NOTE: ONLY WIRING NECESSARY TO THE DESCRIPTION IS SHOWN.

Figure 10-15 Mounting Panel Wiring

Customer Designed Interfaces

The customer who elects to design his own interface package must provide both electrical and mechanical interfaces. In addition to the requirements of the following section, the user should keep in mind the following definitions:

1. External bus signals are positive pulses or positive levels, allowing direct TTL-logic interfacing with appropriate diode clamp protection.

2. These positive pulses and levels change from ground (0V to 0.4V) to a positive voltage between 2.4V and 3.6V.
3. All signal lines are loaded within the two external bus interfaces in the PDP-8/E. Signals coming from the peripheral are inactive when a voltage potential is applied to them; conversely, signals going to the peripheral are inactive when no voltage potential is applied to them.

Figure 10-16 provides logic diagrams of the circuits to which the user must interface. The details of the method used are left to the customer.

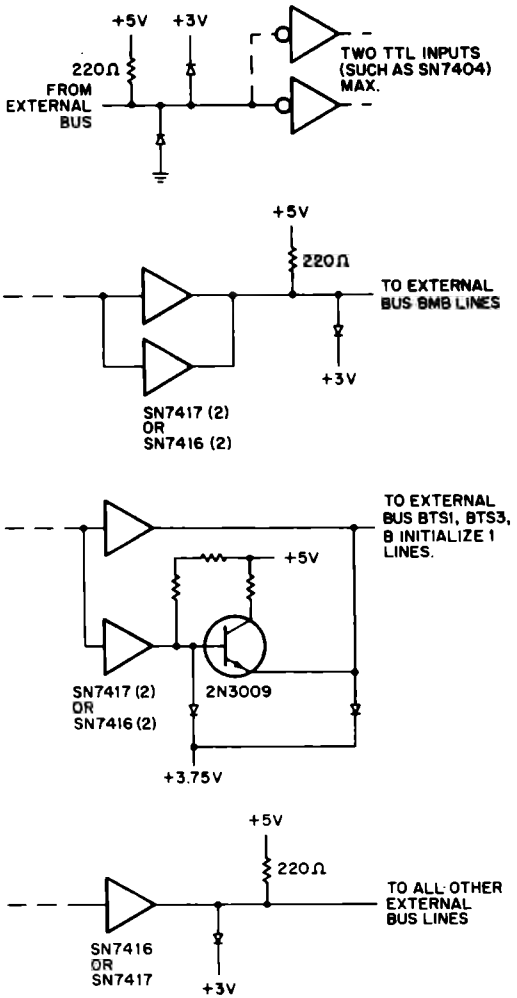


Figure 10-16 Interface Logic Diagrams

RESTRICTIONS AND CRITERIA

Cooling

The low power consumption of M-series modules results in a total dissipation of about 15W in a typical H911 mounting panel containing 64 modules. Convection cooling is sufficient for a few mounting panels, but forced air cooling should be used when a very large system is built.

Signal Terminating

Termination is required on Positive I/O Bus interface cables longer than 20 feet, and may be desirable on shorter cables. The following signals should be shunted to ground by a 100 ohm resistor: IOP1; IOP2; IOP4; BTS1; BTS3; B INITIALIZE 1. If a series of peripherals is being used, the termination is inserted in the last peripheral. A DEC G717 resistor terminator module is available for this purpose.

Timing Criteria

Timing criteria must be considered only when peripherals having high operating speeds (over 5kHz) are being interfaced. The following information concerning interrupt processing must be understood.

a. The interrupt feature must be turned on via the ION instruction in order for the device to be allowed to interrupt the processor.

b. In order to honor the interrupt, the central processor must have completed the instruction it is presently doing.

c. When an interrupt request is honored, the hardware of the machine executes an effective JMS to location 0 in memory field 0, and also disables the interrupt system.

d. An interrupt servicing routine must be resident in memory and the starting address of this routine must be defined in the memory location immediately following location 0.

The longest time required to honor an interrupt request is approximately the time duration of the slowest instruction. Thus, for a PDP-8/E without the EAE option, this time would be 4.3 microseconds (the time required to complete a 3-cycle instruction). For the PDP-8/E with the EAE option, the time would be 9.8 microseconds. These times assume an interrupt request just after the processor enters the FETCH state.

The following examples illustrate the use of this timing criterion.

EAMPLE 1—PDP-8/E without EAE option.

Time between interrupts	50.0 μs
Maximum processor time before interrupt	−4.3 μs
Time for hardware JMS to location 0	−1.4 μs
Maximum time allowed for servicing before possible error arises	44.3 μs

EXAMPLE 2—PDP-8/E with EAE option.

Time between interrupts	50.0 μs
Maximum processor time before interrupt (with EAE option installed—24-bit long shift)	−9.8 μs

Time for hardware JMS to location 0	:	−4.3 μs
Maximum time allowed for servicing before possible error arises	:	35.9 μs

Another timing criterion is concerned with peripheral gating time from IOP to SKIP, from IOP to AC input signals, and from IOP to AC CLEAR. To avoid time delay problems, these gating times must be limited to 100 nanoseconds.

The third timing criterion is concerned with the delays inherent in interconnecting cabling. DEC logic generates waveforms with rising edges containing frequencies of over 100 MHz. At these frequencies the inductance, mutual inductance, capacitance, and tramission line properties of the external bus cabling become significant. To avoid problems, consider the following when interfacing.

a. The propagation delay of typical wiring (1.5 nanoseconds/ft) is often significant when overshoot and reflections are considered.

b. The current carrying capacity of a wire is only V/Z(0) until the wave has propagated along the wire three times. Typical wiring has a characteristic impedance of approximately 150 ohms, so that the current available at the end of the wire for rising waveforms is only 20 milliamps until reflections have propogated, regardless of the source current available.

c. The inductance and capacitance of wiring combine to produce high frequency ringing on the transitions of waveforms. This ringing can be controlled by resistively terminating the line with approximately 100 ohms.

d. The mutual inductance and capacitance of the wiring causes high-frequency crosstalk which may produce false operation of the logic. This crosstalk can be reduced in one of the following ways; minimizing the number of high frequency signal components by clipping or clamping high-frequency ringing with a level terminator circuit, or wiring with short wires and/or twisted pairs, thereby reducing coupling. Clamping can also be used to prevent the excursion of the output or input voltages beyond certain predetermined limits. This is sometimes necessary to prevent false triggering or electrical damage to gates.

CABLING RULES AND SUGGESTIONS

1. Round and flat coaxial cable are electrically interchangeable and may be intermixed in a system. If cables will be subjected to extraordinary abuse, round coaxial cable is preferable when connecting free-standing cabinets.

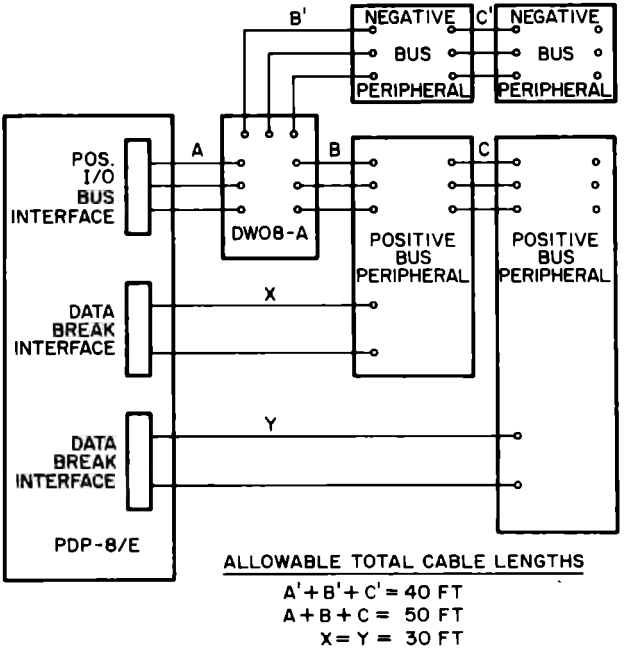
2. Indiscriminate mixing of shielded flat cable and coaxial cable is not advised. DEC recommends that all cables be shielded flat mylar cables, except when the user is trying to gain maximum length or is connecting free-standing cabinets. Not more than one change from coaxial to flat cable, or vice-versa, should be made over the length of a bus.

3. The following cable length restrictions should be observed.

CABLE	TYPE	MAXIMUM LENGTH
1, 2, and 3 from the Positive I/O Bus interface	Coaxial	50 ft
	Flat	45 ft

If a DW08-A I/O converter panel is connected onto cables 1, 2, and 3, the system can accommodate negative bus peripherals. The normal positive bus maximum cable lengths remain as indicated; the maximum cable lengths for the converted bus (negative) are 10 feet shorter than that indicated. See Figure 10-17.

1 and 2 from the Data break interface	Coaxial	30 ft
	Flat	25 ft



NOTE: LENGTHS GIVEN FOR COAXIAL CABLE; EACH IS A MAXIMUM ALLOWABLE LENGTH.

Figure 10-17 Maximum Bus Lengths

4. DEC cable has the nominal characteristics listed below. The user should ensure that whatever cable he uses exhibits approximately the same characteristics.

Z = 95 + or − 5 ohms
C = 13.75 pF/ft (unterminated)
L = 124 nH/ft
R = 0.095 ohm/ft
V (p) = 1.5 ns/ft

5. The cables supplied with the Positive I/O Bus interface and the Data Break interface can be obtained in standard length, as outlined in table 10-3.

Table 10-3 Table Of Standard Cable Lengths

CABLE	LENGTH
BC08J-6	6 feet
BC08J-10	10 feet

M-SERIES MODULE SUMMARY

The following is a list of M Series modules available from Digital Equipment Corporation that can be used in designing special interfaces and special devices. The majority of these modules are described in DEC's DIGITAL LOGIC HANDBOOK. For modules that cannot be found in the handbook, contact the nearest Digital representative for information.

TYPE	FUNCTION	DESCRIPTION
M002	15 Loads	Fifteen +3V sources each capable of driving 10 unit loads. Can be used for tying off unused inputs.
M040	Solenoid Driver	Output ratings of −70V and 0.6A allow these 2 drivers to be used with a variety of medium current loads.
M050	50 ma Indicator and Driver	Output ratings of −20V and 50 milliamps allow any of the 12 circuits on this module to drive a variety of incandescent lamps. These drivers can also be used as slow speed open collector PNP level shifters to −3V system.
M101	Bus Data Interface	Fifteen two-input NAND gates with one input of each gate tied to a common line. For use in strobing data from the PDP-8/I, PDP-8/L, or PDP-8/E I/O bus. Pins are compatible with the M111 module.
M103	Device Selector	Diode gate, buffering and clamping circuits necessary to decode IOT's from the PDP-8/I, PDP-8/L, or PDP-8/E positive bus. Output pulses are not regenerated, only buffered.
M111	Inverter	Sixteen inverter circuits with a fan-in of 1 unit load and fan-out of 10 unit loads.

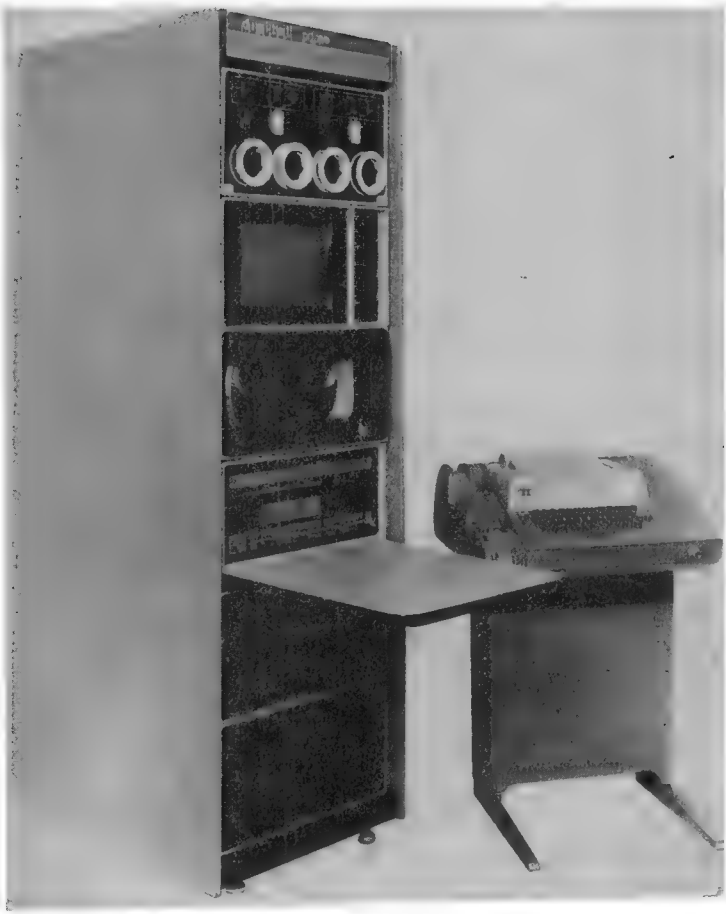
TYPE	FUNCTION	DESCRIPTION
M112	NOR Gate	Ten positive NOR gates with a fan-in of 1 unit load and fan-out of 10 unit loads.
M113	Ten 2-Input NAND Gates	Ten 2-input positive NAND gates with a fan-in of 1 unit load and fan-out of 10 unit loads.
M115	Eight 3-input NAND Gates	Eight 3-input positive NAND gates with a fan-in of 1 unit load and a fan-out of 10 unit loads.
M117	Six 4-Input NAND Gates	Six 4-input positive NAND gates with a fan-in of 1 unit load and a fan-out of 10 unit loads.
M119	Three 8-Input NAND Gates	Three 8-input positive NAND gates with a fan-in of 1 unit load and a fan-out of 10 unit loads.
M121	AND/NOR Gates	Six gates that perform the positive logic function AB + CD. Fan-in on each input is 1 unit load and gate fan-out is 10 unit loads.
M141	NAND/OR Gates	Twelve 2-input positive NAND gates that can be used in a wired OR manner. Gates are grouped in a 4-4-3-1 configuration with a fan-in of 1 unit load and a fan-out that depends on the number of gates ORed together.
M160	Gate Module	Three general purpose multi-input gates that can be used for system input selection. Fan-in is 1 unit load and fan-out is 10 unit loads.
M161	Binary to Octal/Decimal Decoder	A binary to 8-line or BCD to 10-line decoder. Gating is provided so that up to 6 binary bits can be decoded using only M161s. Accepts a variety of BCD codes.
M162	Parity Circuit	Two circuits, each of which can be used to generate even or odd parity signals for four bits of binary input.
M169	Gating Module	Four circuits that can be used for input selection. Each circuit is of an AND/OR configuration with four 2-input AND gates.
M202	Triple J-K Flip-flop	Three J-K flip-flops with multiple input AND gates on J and K. Versatile units for many control or counter purposes. All direct set and clear inputs are available on module pins.

TYPE	FUNCTION	DESCRIPTION
M203	Set-Reset Flip-flops	Eight single input set-reset flip-flops for use as buffer storage. Each circuit has a fan-in of 1 unit load and a fan-out of 10 unit loads.
M204	Counter-Buffer	Four J-K flip-flops that can be interconnected as a ripple or synchronous counter or used as general control elements.
M206	Six Flip-flops	6 D-type flip-flops that can be used in shift registers, counters, buffer registers, and general purpose control functions.
M207	Flip-flops	Six single-input J-K flip-flops for use as shift registers, ripple counters, and general purpose control functions.
M208	Buffer Shift	An internally connected 8-bit buffer or shift register. Provisions are made for gated single-ended parallel load, bipolar parallel output, and serial input.
M211	Binary Up/Down Counter	A 6-bit binary up/down ripple counter with control gates for direction changes via a single control line.
M212	6-Bit Shift Register	An internally connected left-right shift register. Provisions are made for gated single-ended parallel load, bipolar parallel output, and serial input.
M213	BCD Up/Down Counter	One decade of 8421 up or down counting is possible with this module. Provisions are made for parallel loading, bipolar output, and carry features.
M230	Binary to BCD Shift Register Converter	One decade of a modified shift register that allows high speed conversion (100 nanoseconds per binary bit) of binary data to 8421 BCD code. System use of this module requires additional modules.
M302	One Shot Delay	Two pulse-or-level-triggered one-shot delays with output delay adjustable from 50 nanoseconds to 7.5 milliseconds. Fan-in is 2 unit loads and fan-out is 25 unit loads.
M310	Delay Line	Fixed tapped delay line with delay adjustable in 50 nanoseconds increments from 50 nanoseconds to 500 nanoseconds. Two digital output amplifiers and one driver are included.

TYPE	FUNCTION	DESCRIPTION
M360	Variable Delay	Continuously variable delay line with a range of 50 nanoseconds to 500 nanoseconds. Module includes delay line drivers and digital output amplifiers.
M401	Clock	A gateable RC clock with both positive and negative pulse outputs. The output frequency is adjustable from 10 MHz to below 100 Hz.
M405	Crystal Clock	Stable system clock frequencies from 1 kHz to 10 MHz are available with this module. Frequency drift at either the positive or negative pulse output is less than 0.01 percent of the specified frequency.
M410	Reed Clock	A stable low frequency reed clock similar to the M452. Stability in the range 10 degrees C to 70 degrees C is better than 0.15 percent. For use with communications systems and available with only standard teletype and data set frequencies.
M452	Variable Clock	Provides 880Hz, 440Hz, and 220Hz square waves necessary for clocking and for the M706 and M707 modules in a 110-baud teletype system.
M501	Schmitt Trigger	Provides regenerative characteristics necessary for switch filtering, pulse shaping, and contact closure sensing. This circuit can be AND/OR expanded.
M502	Negative Input Converter	Pulses as short as 35 nanoseconds can be level shifted from —3V systems to standard M-Series levels by the two circuits in this converter. This module can also drive low impedance terminated cables.
M506	Negative Input Converter	This converter levels shift pulses as short as 100 nanoseconds from —3V systems to M-Series levels. Each of the 6 circuits on this module has a 10 milliamp load resistor on the negative input.
M507	Bus Converter	Six inverting level shifters that accept —3V and GND as inputs and have an open collector NPN transistor at the output. Output rise is delayed by 100 nanoseconds for pulse spreading.

TYPE	FUNCTION	DESCRIPTION
M516	Positive Bus	Six 4-input NOR gates with overshoot and undershoot clamps on one input of each gate. In addition, one input of each gate is tied to +3V with the lead brought out to a connector pin.
M602	Pulse Generator	The two pulse amplifiers in this module provide standard 50 nanoseconds or 110 nanoseconds pulses for M-Series systems.
M617	Six 4 Input NOR Buffers	Six 4-input positive NOR gates with a fan-in of 1 unit load and a fan-out of 30 unit loads.
M627	Power Amplifier	Six 4-input high speed positive NAND gates with a fan-in of 2.5 unit loads and a fan-out of 40 unit loads.
M650	Negative Output	The three non-inverting level shifters on this module can be used to interface the positive levels or pulses (duration greater than 100 nanoseconds) of K- and M-Series to −3V logic systems.
M652	Negative Output Converter	These two circuits provide high-speed, non-inverting level shifting for pulses as short as 35 nanoseconds or levels from M-Series to −3V systems. The output can drive low impedance terminated cables.
M660	Positive Level Driver	Three circuits provide low impedance, 100-ohm, terminated cable driving capability, using M-Series levels or pulses of duration greater than 100 nanoseconds. Output drive capability is 50 milliamps at +3V or ground.
M661	Positive Level Driver	Three circuits provide low-impedance unterminated cable driving. Characteristics are similar to M660 with the exception that +3V drive is 5 milliamps.
M730	8 Bus Positive Output Interface	General-purpose positive bus output module for use in interfacing many positive level (0 to +20V) systems to the PDP-8/I, PDP-8/L, or PDP-8/E. Module includes device selector, 12 bit parallel output buffer, and adjustable timing pulses.
M731	8 Bus Negative Output Interface	Identical to M730, except that outputs are level shifted for 0 to −20V systems.

TYPE	FUNCTION	DESCRIPTION
M732	8 Bus Positive Input Interface	General purpose positive bus input module for interfacing many positive level (0 to +20V) systems to the PDP-8/I, PDP-8/L, or PDP-8/E. Module includes device selector, 12 bit parallel input buffer, and adjustable timing pulses.
M733	8 Bus Negative Input Interfacer	Identical to M732 except that inputs are level shifted from negative voltage systems.
M901	Flexprint Cable Connector	Double-sided 36 pin shielded mylar cable connector. All pins are available for signals or grounds. Pins A2, B2, U1, and V1 have 10 ohm resistors in series.
M902	Resistor Terminator	Double-sided 36 pin terminator module with 100 ohm terminations on signal leads. Alternate grounds are provided as in the M903 and M904.
M903	Connector	Double-sided 36 pin shielded mylar cable connector with alternate grounds for I/O bus cables.
M906	Cable	Eighteen load resistors clamped to prevent excursions beyond +3V and ground. This terminator can be used in conjunction with the M623 to provide cable driving ability.



PDP-8/E Expanded System

CHAPTER 11

INSTALLATION AND PLANNING

SPACE REQUIREMENTS

Adequate space must be provided at the installation site to accommodate the PDP-8/E and related peripheral equipment and to allow access to all doors and panels for maintenance.

The PDP-8/E is available in two configurations, rack-mounted and table-top. Each offers many advantages that should be considered when planning the type of system desired.

The table-top PDP-8/E (see Figure 11-1) requires much less space than the rack-mounted version. The user may, if he desires, place the table-top model on a corner of his desk and operate the computer right in his office. This places the control panel within easy reach of someone seated at the desk. No special cooling is required, and, in most cases, a standard office electrical outlet provides adequate power. This table-top version is approximately 10½ in. high, 19 in. wide and 24 in. deep. It weighs approximately 100 pounds, and is made up of the central processor, memory, console, power supply, and chassis assembly with top cover. The chassis contains an OMNIBUS, an H724 Power Supply, a 15-foot Power Cord, and a Table-Top cover.

Expansion capability is included within the box for many additional options that can be included or added at a later date without requiring additional space. The basic machine occupies 9 OMNIBUS slots. Up to 11 additional modules can be added on the basic OMNIBUS. Use of the OMNIBUS Expander permits up to 18 additional modules to be added, providing a total of 38 slots. (Two slots are used to interconnect the OMNIBUS and 9 slots are used for the basic system leaving 29 slots available for options.) The system can be further expanded with the addition of the Expander Box, which provides slots for 18 or 36 additional modules.

The System Expander Box, type BA8-A, includes one KC8-EB Blank Front Panel, one Power Chassis Assembler, one OMNIBUS, and a BC08H Cable Set (cover is included). One additional OMNIBUS Expander can be added to increase Bus capability.

The rack-mounted PDP-8/E (see Figure 11-2) can be installed in a DEC cabinet or mounted in a customer cabinet. The Chassis contains an H919 Bus Assembly, an H724 Power Supply, a 15 ft. Power Cord, and Chassis Slides. The rack-mounted version is approximately 10-½ in. high, 19 in. wide, and 23-¼ in. deep, and weighs approximately 90 pounds.

Expansion capability of the rack-mounted PDP-8/E is the same as that of the table-top model. The rack-mounted System Expander Box, type BA8-B, includes a DC8-EB Blank Front Panel, a Power Supply, Chassis Assembly with one OMNIBUS, and a BC08H Cable Set (rack-mountable

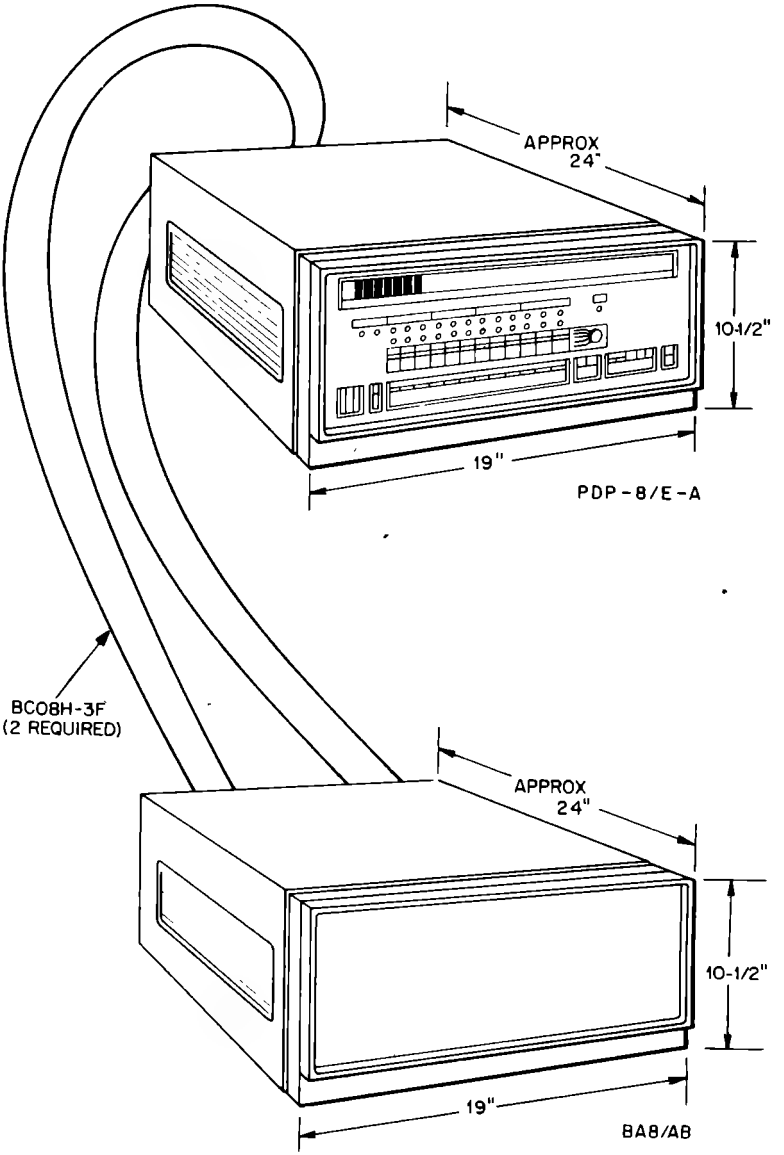


Figure 11-1 Table-Top PDP-8/E System and Expander Box

slides are included). An additional OMNIBUS Expander can be added to increase the Bus capabilities.

This cabinet arrangement allows the user to place the processor and an assortment of peripherals in one area. The basic cabinet (see Figures 11-3 and 11-4) is a Type H960B. Additional option cabinets (Type H961A) can be purchased and connected to the basic cabinet. Each cabinet is approximately 71-7/16 in. high, 21-1/16 in. wide, and 30 in. deep. The door swing is approximately 22 inches (both front and back doors). Thus, overall space requirement for each cabinet is approximately 22-in. by 74-in.

The standard Teletype Model ASR33 requires floor space approximately 22-1/4 in. wide by 18-1/2 in. deep. Standard signal cable length restricts the location of the Teletype to within eight feet of the computer equipment cabinet.

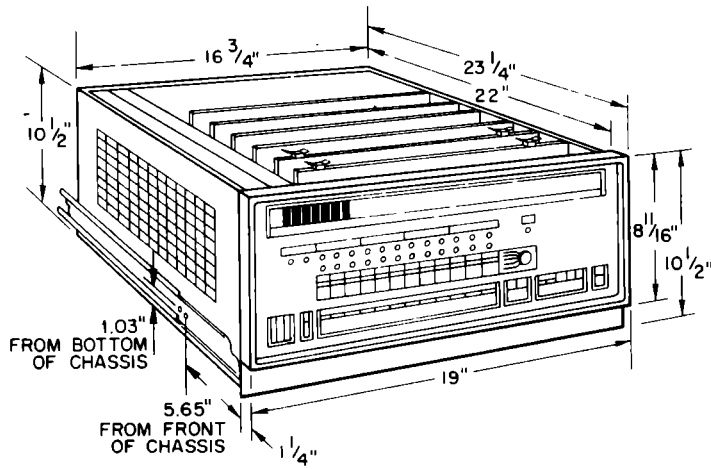


Figure 11-2 Rack-Mounted PDP-8/E Dimensions

Digital Equipment Corporation manufactures a standard 19-in. mounting frame assembly that offers the customer complete flexibility in selecting hardware to design the cabinet. It is a complete enclosure designed to house module racks, power supplies, computer systems, and peripherals.

The frame assembly, which includes a filter cover, is designed for sophisticated computer systems. It is constructed of rugged 12- and 13-gauge steel. The two pairs of frame uprights have 9/32 in. holes grilled at standard EIA spacings (5/8-5/8-1/2) the full length of the 63-in. mounting panel height.

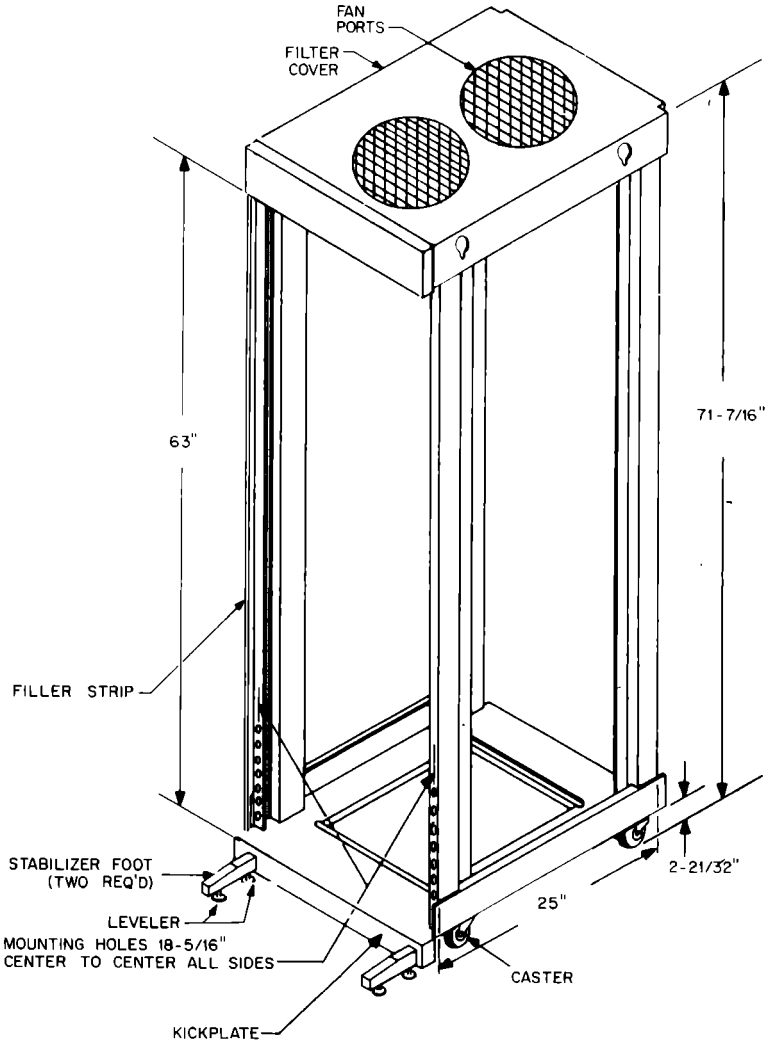


Figure 11-3 Typical H960-Series Cabinet Frame—Dimensions

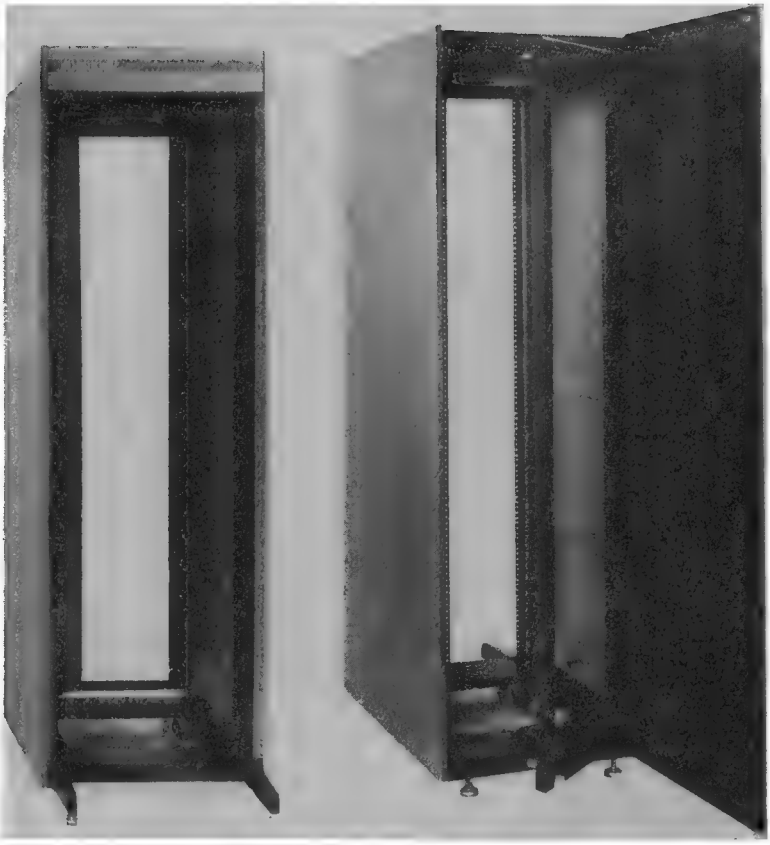


Figure 11-4 DEC H960B Cabinet

ENVIRONMENTAL REQUIREMENTS

Ambient temperature at the installation site can vary between 30 degrees F and 130 degrees F (between 0 degrees C and 55 degrees C) with no adverse effect on computer operation. To extend the life expectancy of the system, however, it is recommended that ambient temperature be maintained between 70 degrees F and 85 degrees F (between 21 degrees C and 30 degrees C). Humidity requirement is from 10 percent to 90 percent without condensation.

During shipping or storing of the system, the ambient temperature may vary between −4 degrees F and 150 degrees F (−20 degrees C and 65

degrees C). Although all exposed surfaces of all DEC cabinets and hardware are treated to prevent corrosion, prolonged exposure to extreme humidity should be avoided.

POWER REQUIREMENTS

PDP-8/E—	Power Dissipation	450 W
	Input Power	95V to 130V 47 to 63 Hz, 6A
	or	
		185V to 250V 47 to 63 Hz, 3A
Teletype—	Input Voltage	104V to 126V, 60Hz + or — 0.45 Hz 207V to 253V, 50Hz + or — 0.50 Hz
	Line Current Drain	2.0A
	Power Dissipation	150 W

Cable Requirements—The PDP-8/E External I/O cable is a combination shield and ribbon cable. (The maximum length of the Data Break Cable, that can be used is 25 feet. The maximum length which can be used on the Positive I/O Bus Interface cables is 45 feet.) The cable between the PDP-8/E and the Expander Box is 3-½ feet long, and cannot be lengthened.

INSTALLATION PROCEDURE

Power Installation—The ac power requirements of the PDP-8/E computer are consistent with good electrical practice. The importance of correct electrical connections cannot be overstressed. Voltage readings must be made at the receptacle before the computer is plugged in, and careful checks must also be made after it is plugged in. It is an extremely wise precaution to take a voltage reading from the frame of the computer to the nearest grounded metal object before touching the computer. Figure 11-5 shows the recommended wall receptacle wiring.

It is generally advisable to provide a separate central load breaker panel for the computer system, with a breaker for the computer and for each peripheral receptacle. DEC recommends that the wiring include a run of No. 4 gauge wire from the computer frame to a substantial earth ground. A large water pipe or a steel building beam is adequate in most instances, although some systems may require a direct connection to a grounding stake or other high-quality earth ground. Significant operational difficulties are likely in the event of either a poor neutral or poor ground circuit.

Voltage readings should be made at each receptacle in the computer power system to ensure adherence to these power requirements. A checkout procedure for testing the electrical system is provided by DEC on request.

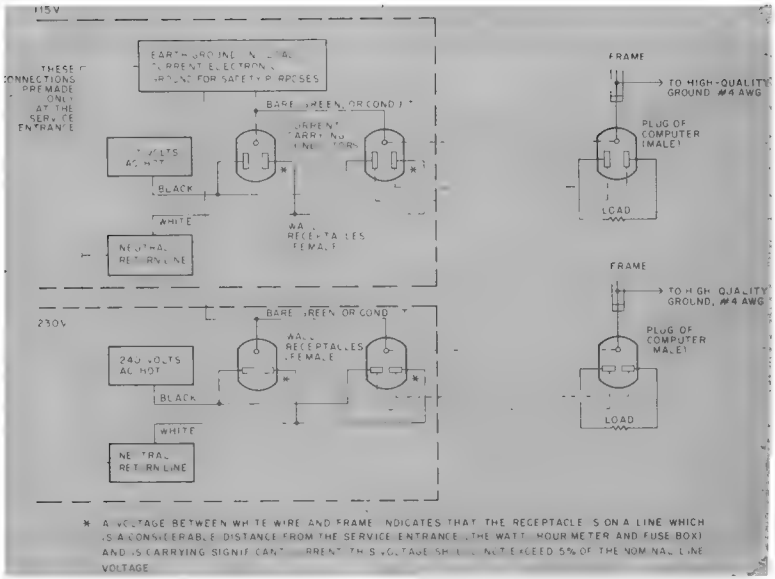

115V
BARE GREEN OR COND.
BLACK
WHITE
NEUTRAL RETURN LINE
230V
240 VOLTS AC HOT
FRAME
TO HIGH-QUALITY GROUND #4 AWG
PLUG OF COMPUTER (MALE)
LOAD
* A VOLTAGE BETWEEN WHITE WIRE AND FRAME INDICATES THAT THE RECEPTACLE IS ON A LINE WHICH IS A CONSIDERABLE DISTANCE FROM THE SERVICE ENTRANCE (THE WATT-HOUR METER AND FUSE BOX) AND IS CARRYING SIGNIFICANT CURRENT. THIS VOLTAGE SHOULD NOT EXCEED 5% OF THE NOMINAL LINE VOLTAGE

Figure 11-5 Wall Receptacle Wiring

System Installation—The PDP-8/E is crated for shipment to the customer site to prevent damage. Installation is provided by DEC personnel at the customer site.

Computer customers may send personnel to instruction courses on computer operation, programming, and maintenance conducted regularly in Maynard, Massachusetts; Palo Alto, California; and Reading, England.

Table 11-1 provides installation data and space requirements which should be considered when installing a PDP-8/E system and related equipment. Figure 11-6 provides the necessary cabinet layout dimensions for those who are installing DEC Cabinet (H960B or H961B).

DEC engineers are available during installation and testing for assistance or consultation. Further technical assistance in the field is provided by home office design engineers or branch office application engineers.

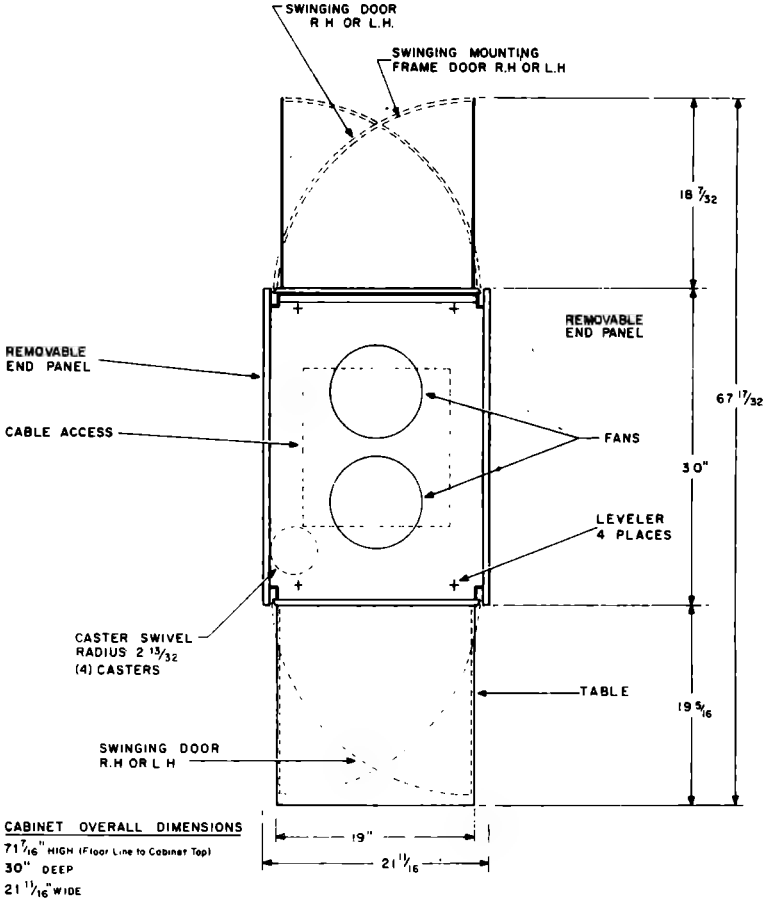


Figure 11-6 Cabinet Layout Dimensions

Table 11-1
Installation Data

Unit	Weight	Dimensions (in.)			Heat Dissipation	115V AC Current Amps		Power Consumption	Mounting Panels	Remarks
	(lbs)	Height	Width	Depth	BTU/Hr.	Nom.	Surge	(KW)		
PDP-8/E Rack-mounted	90	10 1/2	19	23 1/4	1,700	5.6 (Max.)	—	0.50	2	
PDP-8/E Table-Top	100	10 1/2	19	24	1,700	5.6 (Max.)	—	0.50	—	
Standard Cabinet H960B (Empty)	120	71 7/16	21 11/16	25		—	—	—	12 (available)	
System Expander Box BA8/AA (rack-mounted)	90	10 1/2	19	23 1/4	1,700	5.6 (Max.)	—	0.50	2	
System Expander Box BA8/AB (Table Top)	100	10 1/2	19	24	1,700	5.6 (Max.)	—	0.50	—	
Teletype ASR-33	70	45	23	19	375	2.0	8.0	0.11	—	
Paper Tape Reader PR8/E	32	10 1/2	19	16	510	1.0	—	0.15	2	
Paper Tape Punch PP8/E	32	10 1/2	19	16	680	2.0	—	0.20	2	
Card Reader CR8/E	47	18	14	20	900	2.7	—	0.30	—	Table space required
Card Reader Optical Mark CM8/E	47	18	14	18	1,000	2.7	—	0.30	—	Table space required
DECtape Control TC08	30	10 1/2	19	—	1,000	3.0	—	0.30	4	Controls up to 4 TU56 Transports
Dual DECtape Transport TU56	80	10 3/8	19	15 1/4	1,740	115V=2.85A +10V=800MA ± 10% −15V =550MA 10%		0.51	2	Optional +5V or +10V supply may be used

Table 11-1
Installation Data (cont)

Unit	Weight	Dimensions (in.)			Heat Dissipation	115V AC Current Amps		Power Consumption	Mounting Panels	Remarks
	(lbs)	Height	Width	Depth	BTU/Hr.	Nom.	Surge	(KW)		
Disk File and Control DF32D, DS32D	75/60	10 1/2	19	23 5/8	1,700	3	3	0.50	2	
Disk File and Control RF08/RS08	depending on size of system	71 7/16	21 11/16	30	510/620 (Plus RS08 Motors)	1.3 2.6 Plus 4.0A per RS08 motor	—	0.15/0.30 (Plus RS08 Motors)	Separate Cabinet	Dimensions are for subsystem mounted in 2 standard cabinets
Disk Cartridge System RK8	500	71 7/16	21 11/16	25	680 (Logic) 2400 (Per Drive)	4 5 (motors & blowers)		0.20 0.70 (Per Drive)	Separate Cabinet	Will handle up to 3 additional RK01 Disk Files
A/D Converter AD01	15	5 1/4	19	15	170	0.45	—	0.05	1	
D/A Converter AA50A	20	5 1/4	19	15	200	0.3	—	0.06	1	
Multiplexer Control AA05/AA07	depending on size of system	71 7/16	21 11/16	30		Depending on size of system		—	Separate Cabinet	Dimensions are for subsystems mounted in a standard cabinet
IDVM AF04A	300	71 7/16	21 11/16	30	2,350	6.0	13.2	0 69	Separate Cabinet	
Line Printer and Control LE8/FA (80 col., 64 ch.)	280	46	24	22 1/4	1,125	30	—	0.33	—	Paper: standard fanfold Edge punched 4" to 9 7/8" wide Paper guide: 5" paper catcher 11 1/2"
Line Printer & Control: LE8/JA (132 col., 64 ch.)	300	46	48	25	1,700	50	—	0.50	—	Paper: 14" to 14 7/8" wide Paper guide: 5"
30" Incremental Plotter & Control XY8/EA	31	10	39 1/2	14 3/4	600	1.5	—	0.175	—	Table space required for Plotter

Table 11-1
Installation Data (cont)

Unit	Weight	Dimensions (in.)			Heat Dissipation	115V AC Current Amps		Power Consumption	Mounting Panels	Remarks
	(lbs)	Height	Width	Depth	BTU/Hr.	Nom.	Surge	(KW)		
12" Incremental Plotter & Control XY8 EB	18	9 3/4	18	14 1/4	600	1.5	—	0.175	—	Table space required for Plotter
Oscilloscope Display: VR03A	17 1/2	6	8 1/2	17	170	0.5	—	0.05	2	Tektronix Model RM602
Oscilloscope Display: VR14	68	10 1/2	19	17	680	1.75	—	0.20	2	
Storage Tube Display: VT01	51	11 7/8	11 5/8	22 3/4	850	2.1	—	0.25	—	Table space required
Joy Stick Cursor Controller H306	3	5 1/16	5	8 5/16	—	—	—	—	—	Table space required

SYSTEM CONFIGURATIONS

Basic Cabinet Configuration—The recommended equipment assignments for the basic cabinet is illustrated in Figure 11-7. The fixed assignment is given to both the Rack-Mounted PDP-8/E and the Expander Box. Plug-in options can be located in either unit with no module assignments. The assigned priorities start at the top and work down. This should save the user a considerable amount of time in planning unit placement. This priority assignment results from a study concerned with the optimum performance of the system.

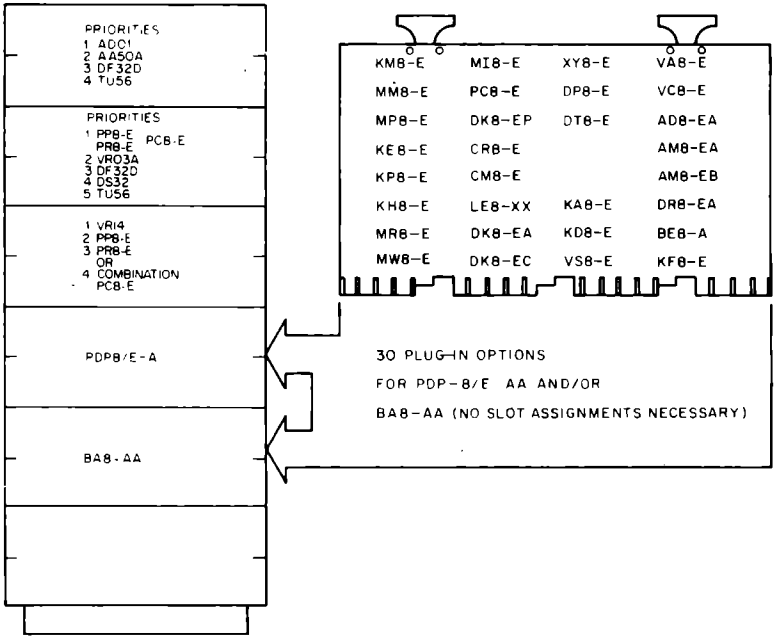
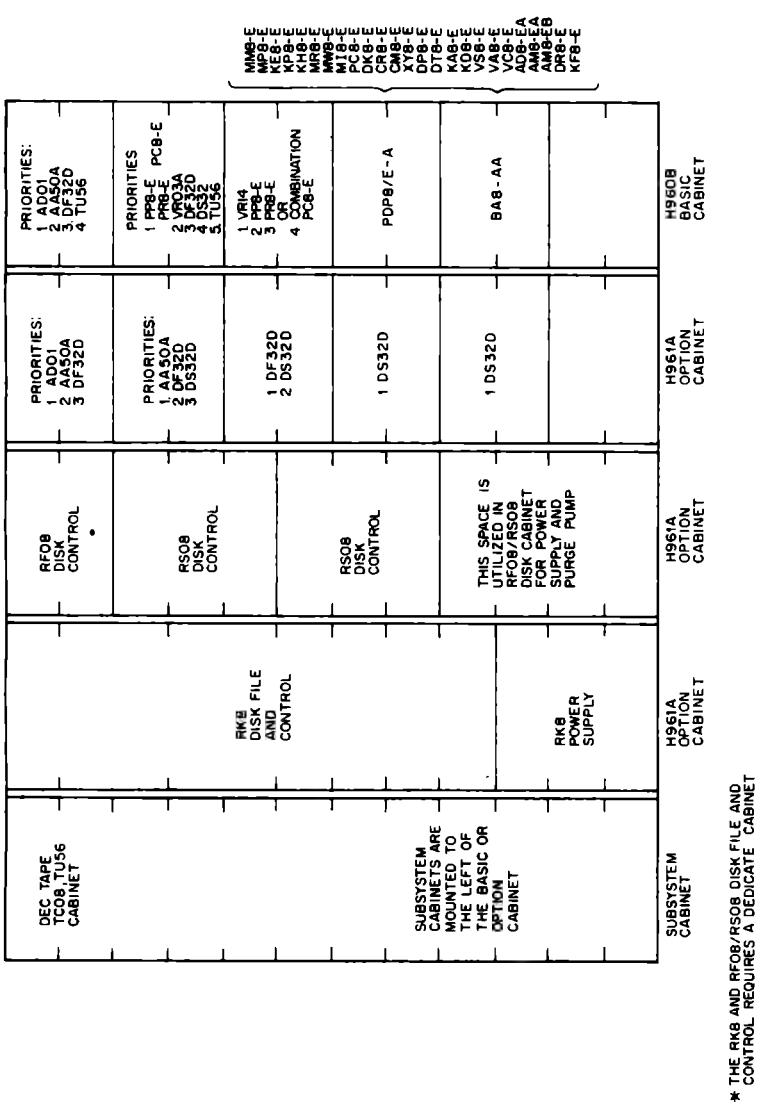


Figure 11-7 Basic Cabinet Configuration

Option Cabinet Configurations—A further expansion of the system is illustrated in Figure 11-8, where many combinations of equipment have been considered. Optional cabinets have been added to the left of the basic cabinet. The equipment priority assignments are shown in an arrangement with associate equipment grouped together. The Subsystem Cabinet is a H961A, which is used when any subsystem type configuration is desired. The RK8 Disk File and control option uses a dedicated cabinet. The RF08/RS08 Disk requires a dedicated cabinet for the RF08 Control and the first two RS08 Disks. A cabinet can be dedicated to the DF32-D disk and control, which will hold up to three DS32-D disk expanders. A cabinet can also be dedicated to DECtape, which will hold a TC08 DECtape control and four TU56 Dual DECtape units.

System Block Diagram—Figure 11-9 illustrates the relationships of the various options with the bus systems and the PDP-8/E Processor.



* THE RK8 AND RF08/RS08 DISK FILE AND CONTROL REQUIRES A DEDICATE CABINET

Figure 11-8 PDP-8/E Cabinet Configurations

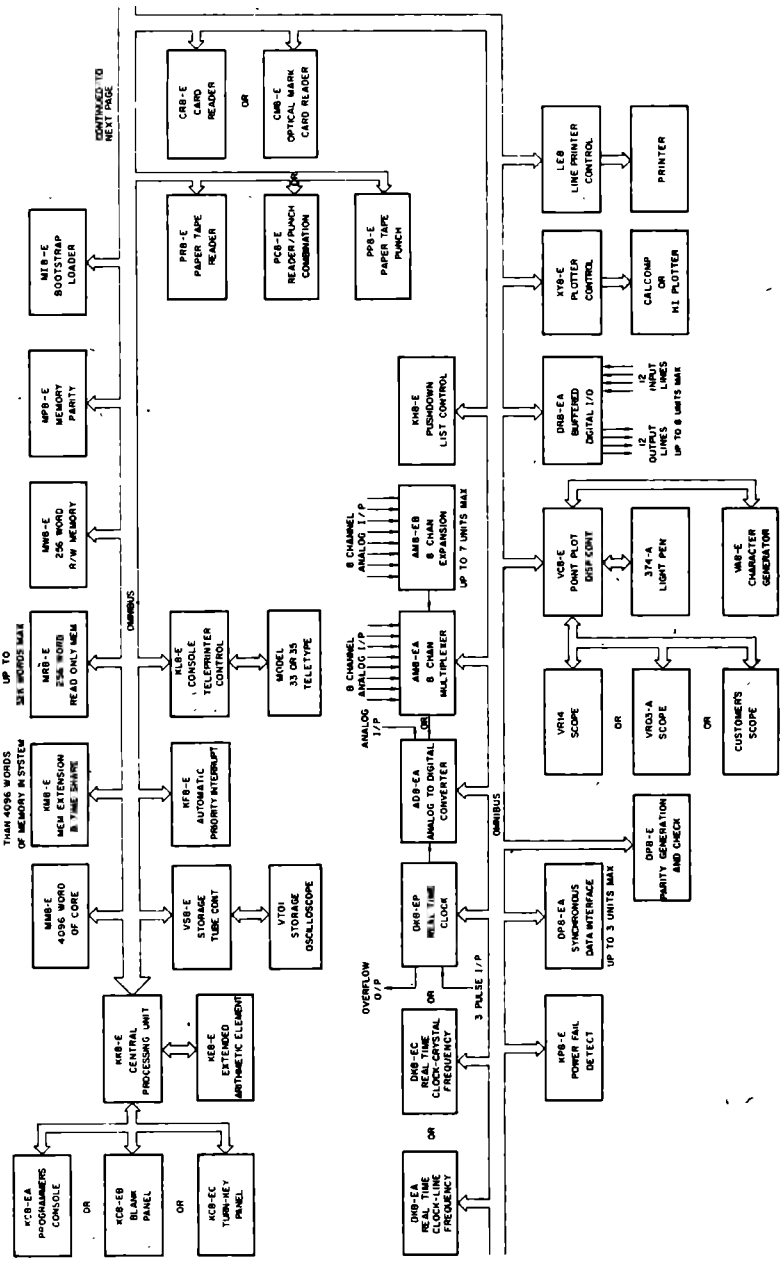


Figure 11-9 PDP-8/E System Block Diagram

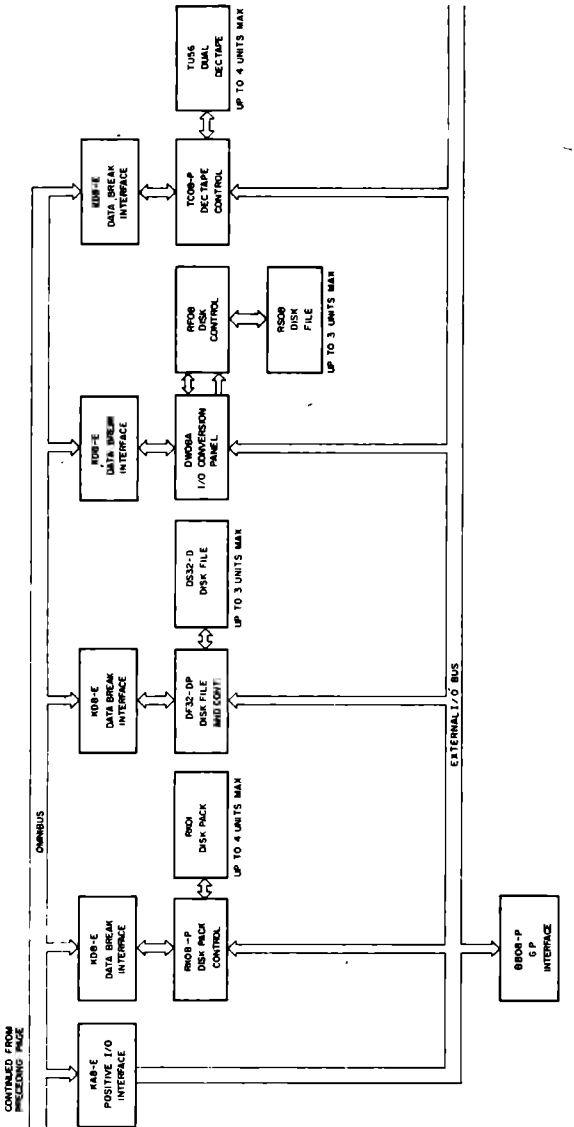


Figure 11-9 PDP-8/E System Block Diagram (Continued)

part 4

DEC SERVICES

CHAPTER 12

EQUIPMENT SUPPORT SERVICES

INTRODUCTION

DEC offers many services to ensure that the equipment is properly operated, properly maintained, and the time between any DEC equipment failure is minimal. Options include maintenance services (depot and field), training and Information Services.

SECTION 1

MAINTENANCE AND SERVICE OPTIONS

Digital Equipment Corporation's Field Service Organization offers a wide range of services to accommodate the needs of DEC equipment users. Customers may choose from a broad selection of service contract options, per-call service, or depot repair maintenance to ensure a high degree of operating efficiency for their DEC equipment.

Service Contracts

Service contracts may be tailored to complement the user's total operation. In addition to providing all the necessary parts, labor, travel, and test equipment required for remedial maintenance, service contracts also ensure system reliability by providing scheduled, systematic preventive maintenance. Planning and especially budgeting are greatly simplified as service contracts permit the user to fill his maintenance needs at a fixed, monthly charge. Customers utilizing DEC service contracts receive response priority over customers relying on other forms of coverage.

Service Contract Options

On-call service contract coverage provides remedial maintenance when DEC is notified of a system malfunction. Preventive maintenance will be scheduled during the period of coverage selected by the user. The principal period of coverage serves as the basic unit from which service coverage may be expanded to parallel a customer's increased operating schedule. The Principal Period of Coverage consists of eight consecutive hours of on-call coverage from 7 a.m. through 6 p.m., Monday through Friday. The user may extend his coverage clockwise from the principal period of coverage by selecting 12, 16, 20, or 24 consecutive hours of on-call coverage. Coverage of 24 consecutive hours, Monday through Friday, begins on Monday of each week at 7 a.m. and terminates on Saturday of each week at 7 a.m.

The Saturday Principal Period of Coverage consists of eight consecutive hours of on-call coverage from 7 a.m. through 6 p.m., Saturday. The Sunday Principal Period of Coverage consists of eight consecutive hours of on-call coverage from 7 a.m. through 6 p.m., Sunday. As with the Principal Period of Coverage, Saturday/Sunday coverage may be extended clockwise to 12, 16, 20, and 24 hours.

In lieu of on-call coverage, an on-site resident engineer may be required, due to the critical nature of an application or the size and complexity

of a particular installation. The services of a resident engineer consist of 40 hours of coverage during the normal work week. In addition, all necessary spare parts, materials, and test equipment are physically stationed at the user's site to further ensure prompt, efficient remedial and preventive maintenance services.

Eligibility for Service Contract Coverage
A pre-service contract inspection will be required for installations which were not under DEC's maintenance responsibility immediately prior to the requested commencement date of the service agreement. All charges associated with this inspection, including travel, labor, and parts, will be billed to the user at the then-prevailing standard DEC rates. A minimum charge is associated with the pre-contract inspection services. No pre-service contract inspection is required for service agreements scheduled to commence immediately after the expiration of a standard DEC warranty or service contract.

Depot Repair
Depot repair facilities have been strategically located throughout the world to enable users of DEC equipment to receive prompt, efficient service on many standard DEC options and processors. In addition to servicing this equipment, depots are also fully equipped to service and completely rebuild, if necessary, Teletype, Models 33ASR, 33KSR, and 35KSR. Furthermore, users of DEC equipment interested in purchasing rebuilt Teletypes, Models 33ASR and KSR, or using their existing Teletypes on a trade-in basis toward the purchase of new machines, are urged to contact their nearest DEC office for further information.

Depots are fully equipped and staffed with highly trained personnel to offer rapid and economical repair services to the DEC customer. At the user's option, a national transportation firm, contracted by DEC, may be used to route equipment to and from the depot facility.

FIELD INSTALLATION OF ADDITIONAL DEC OPTIONS
Customers wishing to expand their existing systems by purchasing additional DEC options may elect to have the installation of the new equipment performed at a fixed rate to facilitate the budgeting and purchasing processes.

SECTION 2

THE DEC TRAINING PROGRAM
Digital Equipment Corporation offers the DEC Training Program to every organization purchasing a computer. The purpose of the program is to prepare future users to program or maintain their DEC equipment.

A trained full-time staff regularly conducts courses at DEC's main facility in Maynard, Massachusetts, Palo Alto, California, and Reading, England. Class sizes are limited. Therefore, early application for enrollment in the desired course is suggested. DEC provides all training materials necessary for each class.

Course Offerings
Courses vary in length, as shown in the following descriptions. Classes

are normally conducted from 9:00 a.m. to 5:00 p.m., Monday through Friday (holidays excepted). In some cases, classes may be conducted in the evening to ensure ample computer time. Special arrangements can be made to have classes conducted at customer sites. Contact DEC

Training customer relations for further information. DEC reserves the right to change schedule and discontinue courses without prior notice.

Course Objectives

Basic Programming—The Basic Programming course is designed to give non-computer-oriented personnel the fundamentals of computer arithmetic and programming concepts necessary to enter any of DEC's small computer hardware familiarization or programming courses.

Programming—The Programming course will give participants a working knowledge of the appropriate:

Machine Language Instruction Set
Programming Techniques
Input/Output Programming
Editor Programs
Assembler Program
On-Line Debugging Programs
Program Library

Hardware Familiarization—The Hardware Familiarization courses will give participants a working knowledge of the appropriate:

Machine Language Instruction Set
DEC Logic and Symbology
Flow Diagrams
Theory of logic operation of:
- Memory Unit
- Central Processor Unit
- Input/Output Section
- Introduction to Diagnostic Routines

Systems Maintenance—The Systems Maintenance courses are specifically designed to train people to maintain their own systems. The courses are patterned after the ones DEC uses to train its own field service people to maintain a given machine and its standard options. The course is approximately 20 percent laboratory.

COURSE DESCRIPTIONS

Basic Programming

COURSE LENGTH: One Week

PREREQUISITES: None

COURSE DESCRIPTION: The course gives non-computer-oriented personnel the background required to enter any of DEC's small computer hardware familiarization or programming courses. The course consists of basic computer concepts, binary and octal num-

bering systems, problem solution, flowcharting, and programming techniques. Basic computer operation includes instruction set, machine language and symbolic programming, use of operator console, I/O programming, and an introduction to problem-oriented language (FOCAL).

PDP-8 Family Programming (Paper-Tape System)
COURSE LENGTH: One week

PREREQUISITE: Basic Programming Course, or equivalent experience

COURSE DESCRIPTION: Course covers general operation of the PDP-8 Family computers, paper-tape system, software to include PAL III Assembler, Symbolic Tape Editor, on-line debugging system (ODT), program interrupt, data break, Program Library.

PDP-8 Family Programming (DECtape/Disk System)
COURSE LENGTH: One week

PREREQUISITES: Paper-Tape Course, or equivalent experience

COURSE DESCRIPTION: Course covers general operation of mass storage system software including Monitor, PAL-D Assembler, Disk/DECtape Editor, Disk/DECtape On-line Debugging Program (DDT/D), Peripheral Interchange Program (PIP), Program Interrupt, Data Break, Disk and DECtape programming.

PDP-8/E Hardware Familiarization
COURSE LENGTH: One week

PREREQUISITE: Background in basic electronics, computer technology, and basic machine language programming

COURSE DESCRIPTION: The course covers the instruction set, central processor including instruction and data flow, memory operation, instruction logic, program interrupt, data break (one and three cycle), I/O hardware, and an introduction to diagnostic programs.

SECTION 3

ECO SERVICE
Another important step in its commitment to customer service has been taken by DIGITAL with the creation of a new customer information service.

The service will keep customers up to date on engineering changes which result from on going in-plant and field evaluation of DEC equipment. Customers will be informed of all ECO's (Engineering Change Orders) through a publication called the DEC-ECO-LOG.

Through implementation of ECO's a customer can maintain a DEC computer system at the latest level of manufacturing specifications. The DEC-ECO-LOG and up-dating pages will be provided at no cost to all customers with DEC maintenance agreements. Other customers may purchase this service.

APPENDIX A

PROGRAM ABSTRACTS

FAMILY-OF-8 PROGRAMS
The PDP-8/E is delivered to the user complete with a selection of more than 1000 system programs and routines which make the full data processing capability of the new computer immediately available to each user and eliminate many commonly experienced initial programming delays.

The programs described in these abstracts come from two sources: past programming effort on previous computer models, and present and continuing programming effort on these same machines plus the PDP-8/E. Thus, the programming system takes advantage of the many man-years of program development and field testing by digital computer users. There are over 8500 Family-of-8 systems already installed in customer facilities.

Many PDP-8/E programs originated with previous Family-of-8 computers and all program documentation listed in this appendix is also applicable to the PDP-8/L and PDP-8/I.

Programs written by users of Family-of-8 computers and submitted to the DECUS Library (DECUS—Digital Equipment Corporation Users' Society) are immediately available to Family-of-8 users. Consequently, users of all Family-of-8 computers can take advantage of continuing program development. The listing of DECUS programs can be obtained from the Program Library, Digital Equipment Corporation, Maynard, Massachusetts 01754.

In addition to the DECUS library, the DEC Program Library has more than 700 programs for the PDP-8 family developed by the Digital Equipment Corporation Programming staff. A complete listing of these available programs can be obtained by writing to the Program Library.

The following abstracts represent the more commonly used system programs. Because new programs are continually being developed, the user should request the latest program listing. Many of the programs are described in DEC Programming Handbooks—"INTRODUCTION TO PROGRAMMING" and "PROGRAMMING LANGUAGES."

SYSTEM PROGRAMS

Monitor Programs

DEC-08-MEXA-D

PS/8 Programming System
The PS/8 Programming System is a software package that provides an executive routine for servicing up to 15 I/O devices. A minimum of 8K of core memory is required for use of the PS/8 Programming System. The user has device-independent access to an assortment of DECtape units, disk units, Teletypes, high-speed paper tape readers and punches,

card readers, line printers, or other devices for which it is possible to write a handling routine in one or two pages.

The user program may call upon the executive routine for several services, including loading device handlers, assigning user-defined logical names for devices to enable symbolic device referencing, looking up input files on these devices, creating and closing variable and fixed-length output files on these devices, and getting and decoding a line of input from the console Teletype to identify input and output files and options.

PS/8 gives the user, in addition to the language processors (8K FOCAL, 8K FORTRAN, 8K PAL-D, and SABR), absolute and relocatable loaders; a Symbolic Editor; CONVERT (a program to provide file compatibility with the 4K Disk Monitor System); PIP (Peripheral Interchange Program); and an invisible ODT (Octal Debugging Technique), which allows the programmer to debug programs without giving up valuable core space.

DEC-D8-SDAB-D

DISK MONITOR SYSTEM

This system consists of a keyboard-oriented monitor, which enables the user to efficiently control the flow of programs through his PDP-8/E, and a comprehensive software package, which includes a FORTRAN Compiler, Program Assembly Language (PAL-D), Edit program (EDITOR), Peripheral Interchange Program (PIP), and Dynamic Debugging Technique (DDT) program. Also provided is a program (BUILDER) for generating a customized monitor according to the user's particular machine configuration (amount of core, number of disks or DECtapes, etc.).

The system is modular and open ended, permitting the user to construct the software required in his environment, and allows the user full access to his disk (or DECtape)—referred to as the system device—for storage and retrieval of his programs. By typing appropriate commands to the monitor, the user can load a program (construct it from one or more units of binary coding previously punched out on paper tape or written on the disk by the Assembler, and assign it core), save it (write it out, with an assigned starting address, on the system device), and later call it (read it back into core from the system device) for execution.

In order to have a complete DISK/DECtape package, the user may order the following in addition to DEC-D8--SDAB-D above:

1. Disk System Builder	DEC-D8-SBAD-PB
2. Disk Editor	DEC-D8-ESAB-PB
3. PIP	DEC-D8-PDAB-PB
4. Disk DDT	DEC-D8-CDDA-PB
5. Disk DDT Driver (ASCII)	DEC-D8-CDDA-PA
6. Disk/DECtape FORTRAN	DEC-D8-AFA(1-6)-PB

DEC-T8-MRFB-D

TSS/8 Time-Sharing Monitor

TSS/8 (Time Sharing System for the PDP-8/E computer) is a general-purpose, stand-alone, time-sharing system. TSS/8 offers each of up to 24 users a comprehensive library of programs that provide facilities

for compiling, assembling, editing, loading, saving, calling, debugging, and running user program on-line. Any of these library programs can be called into use by typing, in response to monitor's invitation (the dot), the command R and the assigned name of the program. For example, • R FOCAL brings the FOCAL program into core from the disk and automatically executes FOCAL so that it begins typing out its initial dialogue.

The heart of this time-sharing system is a complex of programs called monitor. Monitor coordinates the operations of the various units, allocates the time and services of the computer to users, and controls their access to the system. By segregating the central processing operations from the time-consuming interactions with the human users, the computer can in effect work on a number of programs simultaneously. The executions of various programs are interspersed without interfering with one another and without detectable delays in the responses to the individual users.

Editor Programs

DEC-08-ESAB-D

Symbolic Editor

The Symbolic Editor allows the user to prepare and edit symbolic tapes on-line in ASCII code with the Teletype and/or high-speed reader/punch. The tedious task of correcting symbolic program tapes using the Teletype off-line is thereby avoided. Proper use of the Symbolic Editor can substantially ease the labor and reduce the number of passes necessary to correct symbolic program tapes. The Editor reads a page, or section, of symbolic tape into a buffer in core storage, where it is available for examination and correction upon keyboard command. The page buffer occupies all of core not taken up by the Editor itself, and has a capacity of approximately 6000_{10} characters. When the Editor has finished reading a page into the buffer, a bell rings to signal the user that he may begin editing. The user may then call for a listing of individual (numbered) lines in any order, and insert desired changes and corrections. In addition, text may be added to the buffer, or inserted between specified lines. Groups of lines or individual lines may be moved or deleted by a single command, or the entire page may be erased if desired. Searches may be made and parts of lines changed without retyping the entire line. Upon keyboard command, the Editor will then either list or punch out the corrected lines or page on paper tape. The Editor can also be used to generate a new symbolic tape by typing new text directly on the keyboard. Errors in typing may be corrected simply by typing a rubout.

Assembler Programs

DEC-08-ARXB-D

8K SABR Assembler

SABR (Symbolic Assembler for Binary Relocatable programs) is an advanced one-pass symbolic assembler for 8K to 32K PDP-8 Family computers. SABR programs are core page independent, allowing programs to be written without regard to the 128-word core page of the computer. SABR automatically generates off-page and off-field references for direct and indirect statements. It also automatically connects instructions on

one page to those that overflow onto the next. The list of available pseudo-ops is extensive, including external subroutine calling, argument passing, and conditional assembly. SABR offers an optional second pass to produce a side-by-side octal/symbolic listing of the assembled program.

The relocatable binary tapes produced by SABR are loaded into any field of core memory and executed with the 8K Linking Loader, as is the comprehensive library of subprograms (also used with 8K FORTRAN). These subprograms may be called by any SABR program. SABR also acts as the second pass of the 8K FORTRAN system.

DEC-08-ASAD-D
PAL III Assembler (4K)
The PAL III Symbolic Assembler is a two-pass assembler which translates symbolic programs written in the PAL III language into binary-coded programs. The assembler offers an optional third pass which produces an octal/symbolic printout and/or punchout of the assembled program. PAL III permits symbolic origins, expressions, and references. Its output is in binary code for use with DEC's standard Binary Loader.

DEC-08-CMAB-D
MACRO-8 Assembler
The MACRO-8 symbolic assembler accepts source programs written in symbolic language and translates them into binary form in two passes. MACRO-8 produces an object program tape (binary), a symbol table (for use with DDT), an octal/symbolic assembly listing, and useful diagnostic messages. MACRO-8 is compatible with PAL III, and has the following additional features: user defined macros; double precision integers, floating point constants; arithmetic and Boolean operators, literals, text facilities, and automatic link generation.

DEC-D8-ASAB-D
PAL-D DISK ASSEMBLER
PAL-D is the symbolic assembly program designed primarily for the 4K PDP-8 family of computers with disk or DECtape. The PAL-D Assembler makes machine language programming easier, faster, and more efficient. Basically, the Assembler processes the programmer's source program statements by translating mnemonic operation codes to the binary codes needed in machine instructions, relating symbols to numeric values, assigning absolute core addresses for program instructions and data. and preparing an output listing of the program, which includes notification of any errors detected during the assembly process. PAL-D incorporates virtually all of the features of both PAL III and MACRO-8.

8K PAL-D DISK ASSEMBLER
Similar to 4K PAL-D, but designed to operate in either 8K or 12K of core memory. Additional pseuo-operators provide features such as pagination of listings, conditional assembly, and binary output control.

Compiler Programs
DEC-08-AFDO-D
4K FORTRAN
The 4K FORTRAN Compiler lets the user express problems in a mixture of English words and mathematical statements. It reduces the time

needed for program preparation and enables users with little or no knowledge of the computer's organization and operating language to write effective programs.

The 4K FORTRAN language consists of four general types of statements: arithmetic, logic, control, and input/output. FORTRAN functions include addition, subtraction, multiplication, division, sine, cosine, arctangent, square root, natural logarithm, and exponentiation.

DEC-08-KFXB-D

8K FORTRAN

The 8K FORTRAN system translates a source program into relocatable binary code. The relocatable binary code is output on paper tape and then loaded into the computer for program execution. The 8K FORTRAN system features: USA Standard FORTRAN syntax; subroutines; two levels of subscripting; function subprograms; input/output supervisors; relocatable output loaded by the 8K Linking Loader; COMMON statements; I, F, E, A, X, and H format specifications; and arithmetic and trigonometric library subroutines.

The 8K FORTRAN system consists of a one-pass compiler, the 8K SABR Assembler, 8K Linking Loader, and a comprehensive Library of subprograms. The system requires a PDP-8 Family computer with at least 8K words of core memory, a Teletype Model 33ASR, and a high-speed paper tape reader and punch. 8K FORTRAN utilizes all available core to 32K. 8K FORTRAN is a modified version of USASI Basic FORTRAN.

DIBOL

DIBOL (Digital Equipment Corporation Business Oriented Language) is a business oriented software system designed for generating and running business applications, programs for billing, accounts receivable, inventory control, payroll, general bookkeeping, and other accounting and business management functions. This software system comprises a simple business oriented language, a data management system to provide input, sorting, and filing facilities, and a monitor program to organize the DIBOL facilities into a unified system.

The software system is divided into two sections, the data section and the procedure section. The data section is used to describe the information used in the program and to allocate memory. The procedure section contains the executable statements. The data management system consists of three parts: generalized input facility, sort, and generalized file update.

The generalized input facility provides a simple means for getting data into the computer. The user specifies with parameter commands his data fields, the desired output record contents, and the input line format. The generalized input facility reads in paper tapes, punched cards, or teletypewriter input; it checks for data errors and creates formatted files on magnetic tape. Some of the errors that can be detected are incorrect data type, missing fields, oversized fields, and illegal commands. The sort program has three phases: internal sort, multiphase merge, and final merge. Control information, which specifies how to arrange a data file, may either be entered through the console or card reader, or read

from magnetic tape. The generalized file update reads transactions from cards or tape, with each transaction coded as insert or delete. The insert code is used either to insert a new item or change an existing item. Through these commands, the transactions are passed against a master file, resulting in an update master tape and a listing of transactions on the line printer.

The DIBOL monitor ties the various DIBOL facilities together into a unified data processing system. By means of a simple interactive dialogue, the monitor executes the subsystems according to operator commands.

After the monitor is loaded, the operator responds to an inquiry by selecting the subsystem to be run. The subsystem can be the compiler, a phase of the sort, or a run-time system. If the run-time system (enabling the computer to execute a DIBOL program) is selected, the operator would then be requested to select an applications program to run.

At the end of the job, control would revert from the applications program to the monitor, which would request the operator to specify the next run.

The minimum configuration for use of the DIBOL system consists of the PDP-8/E, 8K of core memory, four DECtape drives, and a Teletype Model 33ASR. Optional equipment includes a paper tape reader or a card reader for data input and a line printer for data output. (If a line printer is not used, a Model 35ASR is recommended, rather than a Model 33ASR).

Interpretive Programs
DEC-08-AJAD-D
FOCAL-8

FOCAL-8 (FOrmula CALculator for the PDP-8 Family computers) is an on-line, conversational interpretive language, used as a tool by students, engineers, and scientists in solving a wide variety of numerical problems. The language consists of short imperative English statements and mathematical expressions in standard notation. FOCAL puts the full calculating power and speed of the computer at the user's fingertips without requiring mastery of the intricacies of assembly language programming; in fact, the user need not know anything about computers.

Using FOCAL-8, programs can be entered from the Teletype keyboard and immediately executed, with the interpretive features editing, compiling, and executing the stored program. FOCAL is available with additional segments (overlay tapes) which provide increased calculating accuracy, more core for large programs by utilizing two fields of core, or graphic display capabilities. Worthy of special mention is the system segment that gives each of seven FOCAL users the power and flexibility of FOCAL from one computer. This time-sharing system permits one FOCAL program to serve several users so that each feels he has the whole system to himself.

DEC-08-ZOSB-D
EDGRIN

EDGRIN (Editor with Graphic Interpreter) is an on-line, interpretive graphic system for use with the PDP-8/E plus a Model 33ASR Tele-

printer keyboard, the paper tape reader, or the Joystick Cursor, and and certain optional I/O devices. EDGRIN accepts inputs from the teleprinter keyboard, the paper tape reader or the Joystick Cursor, and uses these inputs to generate displays.

EDGRIN source language programs simplify the task of generating displays. The basic graphic functions of character generation, oscilloscope control, coordinate address handling, point plotting, and drawing linear vectors, circles, and arcs are complemented by program control and special function calling features.

Included with EDGRIN is an Editor Translator, which is a general purpose symbolic editor with oscilloscope display capabilities. The Editor Translator provides more than 20 commands and special symbols to input, manipulate, display, and output alphanumeric text strings. The Editor Translator program is furnished in binary coded format on punched paper tape.

DEC-T8-KJZA-D
BASIC-8
BASIC-8 runs under control of the TSS/8 Time-Sharing System and is intended for use by students, scientists, and engineers for writing and running programs involved with the solving of algebraic problems on the computer. It is particularly useful for repetitive calculations. Program statements are written in English and mathematical statements are written in standard mathematical notation with some substitution of symbols.

The BASIC-8 provides service for many users with relatively small programs; however, there are no general restrictions on program length. If a program becomes too large to store in memory and execute at one time, the system stores part of the program on the disk or DECtape, and runs each part separately. This automatic segmentation process is one of the advanced features of the BASIC-8.

STAND-ALONE BASIC
STAND-ALONE BASIC is used independently of the TSS/8 Time-Sharing System and requires 4K of core memory and a DF-32, RF08, or a DECtape unit. The DECtape must be either a TU55 or TU56 unit. Optional output devices such as the CM-8E card reader or the LP08 line printer can be used.

STAND-ALONE BASIC is an interactive program which also is capable of batch mode operation. It is used for writing and running programs involved with the solving of algebraic problems on the computer.

Debugging Programs
DEC-08-CDDB-D
DDT-8 (Dynamic Debugging Tape)
DDT-8 provides a means for on-line program debugging at the symbolic or mnemonic level. By typing commands on the console teleprinter, memory locations can be examined and changed, program tapes can be inserted, selected portions of the program can be run, and the updated program can be punched.

DEC-08-COCO-D

ODT-8 (Octal Debugging Technique)

ODT-8 (Octal Debugging Technique) is a debugging aid which facilitates communication with, and alteration of, the program being run. Communication between operator occurs via the Teletype, using defined commands and octal numbers. ODT-8 is a subset of DDT-8 and occupies three pages of core storage.

The program may be relocated to occupy any three consecutive pages of core.

Loader Programs

DEC-08-LBAA-D

Binary Loader

The Binary Loader is a short routine for reading and storing information contained in binary-coded tapes, using the Model 33ASR Perforated Tape Reader or the High-Speed Perforated Tape Reader.

The Binary Loader accepts tapes prepared by the use of PAL (Program Assembly Language) or MACRO-8. Diagnostic messages may be included on tapes produced when using either PAL or MACRO. The Binary Loader ignores all diagnostic messages.

DEC-08-LHAA-D

"HELP" Loader

The "HELP" Loader loads the standard version of the RIM and BIN Loaders into the PDP-8/E, in less than 90 seconds, replacing manual procedures which require several minutes.

DEC-08-LRAA-D

Read-In Mode (RIM Loader)

The RIM Loader is a minimum routine for reading and storing information contained in read-in-mode coded tapes via the Model 33ASR Perforated Tape Reader.

DEC-08-LUAA-D

TC01 Bootstrap Loader

This is a bootstrap for loading the PDP-8 DECtape Library System designed for use with DECtape Control Type TC01 with TU55 Tape Transports.

Utility Programs

DEC-08-FFAC-D

Math Routines Manual

The following routines are described in the Math Routines Manual.

Square Root Subroutine-Single Precision
: Forms the square root of a single-precision number. An attempt to take the square root of a negative number will give 0 for a result.

Signed Multiply Subroutine-Single Precision
: Forms a 22-bit signed product from 11-bit signed multiplier and multiplicand.

Signed Divide Subroutine-Single Precision

This routine divides a signed 11-bit divisor into a signed 23-bit dividend, giving a signed 11-bit quotient and a remainder of 11 bits with the sign of the dividend.

Double-Precision Multiply Subroutine-Signed

This subroutine multiplies a 23-bit signed multiplicand by a 23-bit signed multiplier and returns with a 46-bit signed product.

Double-Precision Divide Subroutine-Signed

This routine divides a 23-bit signed divisor into a 47-bit signed dividend and returns with a 23-bit signed quotient and a remainder of 23 bits with the sign of the dividend.

Sine Routine-Double Precision

The double-precision sine subroutine evaluates the function Sin (X) for $-4 < X < 4$ (X being in radians). The argument is a double-precision word, 2 bits representing the integer part and 21 bits representing the fractional part. The result is a 23-bit signed fraction $-1 < \text{Sin}(X) < 1$.

Cosine Routine-Double Precision

This subroutine forms the cosine of a double-precision argument (in radians). The input range is $-4 < X < 4$.

Four-Word Floating-Point Package

This is a basic floating-point package that carries data as three words of mantissa and one word of exponent. Common arithmetic operations are included as well as basic input/output control. No functions are included.

Logical Subroutines

Subroutines for performing the logical operations of inclusive and exclusive OR are presented as a package.

Shift Right, Shift Left Subroutines (Single and Double Precision)

Four basic subroutines, shift right and shift left, each at both single and double precision are presented as a package.

Logical Shift Routines

Two basic subroutines, shift right at both single and double precision, are presented as a package. The shifts are logical in nature.

DEC-08-PMPO-D

Read-In Mode (RIM) Punch

The RIM Punch program provides a means of punching out information contained in selected blocks of core memory as RIM-coded tape via the Model 33 ASR Perforated Tape Punch or High-Speed Punch. The punch program may occupy either low or high memory, depending on the version used.

Digital-08-USBO-D

Multianalyzer Display and Analysis

The two-dimensional pulse-height analysis reads in and analyzes two-parameter energy and spectra data. The program receives and executes commands from the keyboard. These commands start and stop data taking, control the displays, and control writing and punching of data. The displays available are: isometric, vertical and horizontal slicing,

differential and integral contours, and "twinkle box". The program is flexible with respect to the dimensions of the data matrix.

DEC-08-YISB-D

Variable Stroke Character Generator for KV8

The Variable Stroke Character Generator (VSCG) is a program which generates and displays the 64 principal ASCII chracters using the KV8 Storage Tube Display Controller. By making use of the redundancy in the English character set, all 64 ASCII characters are described by seven vector coordinate masks plus two control words per character. This is quite different from the conventional stroke vector generators, where all ASCII characters are generated from one standard mask, for example, the starburst pattern.

Characters are generated by executing stroke vectors on the storage tube. Each vector is specified by the address of its end-point coordinates. Execution time is minimized by executing only the required stroke vectors for a particular character.

In addition to displaying characters, VSCG provides control functions for carriage return/line feed, "home" to top left of the scope, horizontal tab, and character stabilization.

VSCG is very versatile and easily modified, as all control functions and characters are table driven.

DEC-08-YPPA-D

Octal Memory Dump

This routine will read the console switches twice to obtain the upper and lower limits of an area of memory, then type on the Teletype an absolute address plus the octal contents of the first four words specified, and repeat this until the block is exhausted, at which time the user may repeat the operation.

DEC-08-YQYB-D

Floating Point Package

The floating point package permits the PDP-8/E to perform arithmetic operations that could not otherwise be done without the addition of costly optional hardware. With the floating point package, the programmer need not concern himself with writing and repeatedly calling complicated arithmetic subroutines. He merely has to specify in one instruction the desired arithmetic operation (operator) and an argument (operand), and the operation will be performed automatically. In addition to increasing accuracy, floating point operations relieve the programmer of scaling problems common in fixed-point arithmetic. The floating point subroutines and interpreter permit the programmer to encode arithmetic operations to either 6 or 10 decimal digits of precision as easily as though the machine had floating point hardware.

DEC-08-YXYA-D

Binary Punch (ASR33 or PP8)

This program provides a means of punching out information contained in selected blocks of core memory as binary-coded tape via the Model 33 ASR Perforated Tape Punch or the High-Speed Punch.

Digital-8-10-U-Sym

BCD to Binary Conversion Subroutine

A basic subroutine for converting binary-coded decimal numbers to their equivalent binary value. Conversion is accomplished by "radix deflation".

Digital-8-11-U-D

Double Precision BCD to Binary Conversion Subroutine

This subroutine converts a 6-digit BCD number to its equivalent binary value in two computer words.

Digital-8-14-U-Sym

Binary to BCD Conversion Subroutine

This subroutine provides the basic means of converting binary data to binary-coded decimal (BCD) data for typeout, magnetic tape recording, etc.

Digital-8-15-U-Sym

Binary to BCD Conversion (Four Digit)

This subroutine extends the method used in Digital-8-14-U-Sym so that binary integers from 0 to 4095 contained in a single computer word may be converted to four binary-coded decimal characters packed in two computer words.

Digital-8-16-S-D

Master Tape Duplicator

This program will duplicate and verify 8-channel paper tapes using a high-speed reader and high-speed punch. The program uses the program interrupt and allows both the reader and the punch to operate at maximum speed. The program accumulates two types of checksums while reading and punching: (1) the number of nonzero characters on the tape, and (2) the sum of characters on the tape (both are taken in modulo 4096).

When duplicating, the program compares the checksums at the end of the tape with the checksums accumulated by the read routine. If these differ, a reader error has occurred and a message is typed.

Digital-8-18-U-D

Alphanumeric Message Typeout

A basic subroutine to type messages packed in computer words. Two 6-bit characters are packed internally in a single word. All Model 33 ASR codes from 301 to 337 and from 240 to 277 (except 243 and 245) can be typed. The typing of line-feed (code 212) and carriage-return (code 215) are made possible by arbitrarily assigning internal codes of 43 and 45, respectively, to represent these characters, thus preventing the output of ASCII code 243 (#) and 245 (%).

Digital-8-19-U-Sym

Teletype Output Subroutines

A group of subroutines useful in controlling Model 33 ASR is presented as a package. Provision is made for the simulation of tabulation stops. The distance "tabbed" may be controlled by the user. Characters whose codes are in groups 241 and 277, inclusive, and 300 through 337, in-

clusive, are legal. Space, carriage return then line feed, and tabulation are provided via subroutines.

Digital-8-20-U-Sym

Character String Typeout
A basic subroutine to type messages stored internally as a string of coded characters. All Model 33 ASR characters are legal.

Digital-8-21-U-Sym

Symbolic Tape Format Generator
The format generator allows the user to create symbolic tapes with formatting. It may be used to condense tapes with spaces by inserting tabs, or merely to align tabs, instructions, and comments.

Digital-8-23-U-Sym

Signed Decimal Print—Single Precision
This subroutine permits the typout of the contents of a computer word as a signed two's complement number. If bit 0 of the computer word is a 1, the remaining bits represent a negative integer in two's complement form; if bit 0 equals 0, the remaining bits represent a positive integer. If the number is negative, a minus sign is printed; if positive, a space.

Digital-8-24-U-Sym

Unsigned Decimal Print, Double Precision
This subroutine permits the typeout of a double-precision integer stored in the usual convention for double-precision numbers. The one exception is that all 24 bits are interpreted as magnitude bits (i.e., bit 0 of the high-order word is not a sign bit). The typeout is in the form of a 7-digit, positive, decimal integer.

Digital-8-25-U-Sym

Signed Decimal Print, Double Precision
This subroutine permits the typeout of the contents of two consecutive computer words as one signed, double-precision, two's complement number. If bit 0 of the high-order word is a 1, the remaining 23 bits represent a negative integer in two's complement form; if bit 0 equals 0, the remaining bits represent a positive integer. If the number is negative, a minus sign is printed; if positive, a space.

Digital-8-28-U-Sym

Single Precision Decimal to Binary Conversion and Typeout ASR33, Signed or Unsigned
This routine accepts a string of up to four decimal digits from the Teletype keyboard and converts it to the corresponding two's complements binary number.

The string may contain as legal characters a sign (+, —, or space) and the digits from 0 through 9. If the first legal character is not a sign, the conversion is unsigned. A back arrow (←) at any point in the string erases the current string and allows the operator to reenter the correct value. Any character after the first, other than another digit or back arrow, causes the conversion to terminate and is found in location SISAVE within the subroutine.

Digital-8-29-U-Sym

Double Precision Decimal to Binary Conversion and Typeout (ASR33), Signed or Unsigned

This routine acepts a string of up to eight decimal digits from the Teletype keyboard and converts it to the corresponding two's complement binary number. The string may contain as legal characters a sign (+, —, or space) and the digits 0 through 9. If the first legal character is not a sign, the conversion is unsigned. A back-arrow (←) at any point in the string erases the current string and allows the operator to reenter the value. Any character after the first, other than another digit or back-arrow, causes the conversion to terminate and is found in location DIDSAV within the subroutine.

DECTAPE SYSTEM SOFTWARE

DEC-08-SUCO-D

DECtape Programming Manual

The DECtape Library System is loaded by a 17_{10} instruction bootstrap routine that starts at 7600_8. This loader calls a larger program into the last memory page, whose function is to preserve on tape the contents of memory from 6000_8-7577_8, and then to load the INDEX program and the directory into those same locations. Since the information in this area of memory has been preserved, it can be restored when operations have been completed. The skeleton system tape contains the following programs:

INDEX—Typing this causes the names of all programs currently on file to be typed out.

UPDATE—Allows the user to add a new program to the files. UPDATE queries the operator about the program's name, its starting address, and its location in core memory.

GETSYS—Generates a skeleton library tape on a specified DECtape unit.

DELETE—Causes a named file to be deleted from the tape.

Starting the skeleton library tape, the user can build up a complete file of his active programs and continuously update it.

DEC-08-EUFB-D

TC01/TU55 DECtape Formatter

The purpose of this system is to record the required timing and mark tracks on a DECtape mounted on the TC01-TU55 DECtape unit.

The program, which never stops, obtains the variable information it needs by communication with the operator via the Teletype Model 33 ASR.

Two full passes are required to complete one DECtape. Upon completion of a sequence, another tape may be mounted and formatted, as the last, without renewed communication between the operator and program. Therefore, marked tape may be produced in great numbers with little operator intervention, at a rapid rate; one tape, excluding set-up time, requires two minutes from start to finish (see also Disk Software).

DEC-08-YPTA-D

DECtape Copy Routine

The DECtape copy program (DTC-8) provides a simple, efficient method for copying one DECtape from another on the PDP-8. Features include the capability of handling nonstandard block lengths (up to 1550_{10} 12-bit words per block), and the ability to reread and validate the copied data, thus ensuring its correctness. DECtape handling routines are all internal and monitor-independent.

Maintenance and Diagnostic Programs

MAINDEC-08-D02B-D

Instruction Test Part 2B

This program is a test of the two's complement add (TAD) and rotate logic (RAL, RTL, RAR, RTR). Random numbers are used in the two's add portion of the test and sequential numbers are used in the rotate portion. Program control depends on operator manipulation of four switches in the SWITCH REGISTER (bits 0, 1, 2, 3). Error information is normally printed out on the keyboard printer.

MAINDEC-08-DO4B-D

Random JMP Test

This program tests the JMP instruction of the PDP-8/E. Most of memory is used as a JUMP field with a random number generator selecting each "jump from" and "jump to" location.

MAINDEC-08-DO5B-D

Random JMP-JMS Test

This is a diagnostic program to test the JMS instruction of the PDP-8/E. Random "from" and "to" addresses are selected for each test. The JMP instruction is tested in that each test requires a JMP to reach the JMS.

MAINDEC-08-DO7B-D

Random ISZ Test

This program is written to test the ISZ instruction of the PDP-8/E. An ISZ instruction is placed in a "from" location, and a "to" location contains the operand. Part 1 of the program selects "from," "to," and operand from a random number generator, with the option of holding any or all constant. Part 2 uses a fixed set of "from," "to," and operand numbers.

MAINDEC-08-DOUA-D

Random ADD-ROTATE Test

This is a diagnostic program to test TAD and rotate instructions. Random numbers and addresses are used to exercise and check computer operation.

MAINDEC-08-D1AC-D

Memory Power On/Off Test

This program is a memory data validity test to be used after a simulated power failure.

MAINDEC-08-D1BO-D

Memory Address Test

The memory address test checks for proper memory address selection on the PDP-8/E.

Extended Memory Checkerboard

The PDP-8/E Extended Memory Checkerboard diagnostic is designed to provide worst-case half-select noise conditions in order to determine the operational status of core memory. Four data patterns and their complements are written and checked for error. The patterns provided generate the worst-case noise conditions for a PDP-8/E equipped with standard or specially purchased core stacks, and test systems equipped with 8K words of core memory. Automatic program selection is provided in order to test all memory stacks from each stack.

Teletype printouts are provided for error identification. Also, the operator is given a degree of control over the program by various SR settings.

MAINDEC-08-D1GB-D

Extended Memory Control

This program tests the extended memory control logic for proper operation. It may be used with a PDP-8/E equipped with a minimum of 4K of extended memory. The program exercises and tests the control IOT's; the ability to reference all fields from field 0, program interrupt and interrupt inhibit, and auto-indexing in each field, and including a special test for the presence of a false memory pulse when a nonexistent memory field is referenced. Errors encountered during running result in a program halt. The halt locations are labeled, and the error may be identified by referencing the program listing or table of error halts.

MAINDEC-08-D1HA-D

Extended Memory Address Test

The PDP-8/E Extended Memory Address Test tests all of memory not occupied by the program to make sure that each location can be uniquely addressed. This consists of a series of four tests. The first two tests write the address and complement address of each memory location into itself, and then check the contents of each location to make sure each is correct. The third test first sets all of memory not occupied by the program to all ones, and then writes a word of all zeroes, except for one bit, into each location and checks for error. The fourth test is similar except that a word of all ones, except for a single bit, is written into each location and checked for error.

MAINDEC-08-D1KA-D

Power Fail Test

This diagnostic is a complete test of the PDP-8/E power fail option with the intervention of the operator.

Basic Memory Checkerboard

The memory checkerboard diagnostic tests memory for core failure on half-selected lines under worst-case conditions. Its use is intended for basic 4K memory systems.

MAINDEC-08-D2EA-D

High Speed Reader Test

This program tests the performance of the High Speed Perforated Tape Reader and Control by scanning a closed-loop test tape for accuracy of transmission. An auxiliary program included with the test punches a tape from which the test loop can be made.

MAINDEC-08-D2GF-D

Family-of-8 High-Speed Punch and Reader Tests

The Family-of-8 High-Speed Reader and Punch Tests are a test package used to test the Type PC02 and PC03 High-Speed Reader Punch when attached to any of the Family-of-8 systems. The tests perform basic input and output control logic tests, reader and punch tests, and reader and punch speed printouts, and provide maintenance loops useful in adjusting the reader and punch.

MAINDEC-08-D2PE-D

Family-of-8 33/35 ASR

Teletype Tests Part 1

The Family-of-8 Teletype 33/35 ASR Tests, Part 1, is the first part of a two-part package used to test the Teletype Model 33 ASR or Model 35 ASR when attached to a Family-of-8 system.

Part 1 contains nine selectable programs numbered from 0 to 10 (octal). The programs are selected by means of the switch register (SR). The programs available are:

PRG0	Basic Input Logic Tests
PRG1	Basic Output Logic Tests
PRG2	Reader Test
PRG3	Test Tape Generator. Punches tape with characters stored in locations 0021 and 0022.
PRG4	Test Tape Generator. Punches Binary Count Pattern test tape.
PRG5	Reader Exerciser. Reads Binary Count pattern tape in random length blocks, and with fixed stalls between characters. The stall is determined at random.
PRG6	Reader Exerciser. Reads Binary Count pattern tape. Fixed stall between characters. Stall count is taken from LOC23.
PRG7	Reader Exerciser. Reads tape punched with any two test characters. Random length blocks and fixed stall between characters. The stall is determined at random.
PRG10	Reader Exerciser. Reads tape punched with any two test characters. Fixed stall between characters. Stall count taken from LOC23.

MAINDEC-08-D2QD-D

Family-of-8 33/35 ASR

Teletype Tests, Part 2

The Family-of-8 33/35 ASR Teletype Tests, Part 2, is the second part of a two-part package used to test the Teletype 33ASR or 35ASR when attached to a Family-of-8 system.

Part 2 contains nine selectable programs numbered from 0 to 10 (octal). The programs are selected by means of the switch register (SR).

The available programs are:

PRG0	Printer Test
PRG1	Punch Test
PRG2	Keyboard Test
PRG3	Combined Reader, Printer, Punch Test
PRG4	Printer Exerciser. Prints lines of characters stored in LOC 0021 and 0022. No stalls.
PRG5	Same as PRG4, but stalls between characters.
PRG6	Punch Exerciser. Punches and read checks data blocks of data stored in LOC 0021 and 0022. No stalls.
PRG7	Same as PRG6, but random stalls between characters punched.
PRG10	Punch Exerciser. Punches and read checks blocks of binary count pattern. Random stalls between characters punched.

MAINDEC-08-D3BC-D

TC01 Basic Exerciser

The TC01 Basic Exerciser is a series of test programs that may be used to gain a high degree of confidence in the data handling ability of a TC01 DECtape Control and one to eight TU55 DECtape Transports. The Basic Exerciser consists of several basic routines that may be individually selected; each routine operates on any configuration of one to eight drives. These routines include a Basic Motion Routine, Search Find All Blocks Test, Basic Search Routine, Start/Stop/Turnaround Test, Basic Write/Read Data Test with eight selectable patterns, and a Parity Generation and Checking Test. The operation of the Basic Motion Routine and the Basic Search Routine are controlled by keyboard input. Also, a Write Data Scope Loop, Read Data Scope Loop, and a Search Scope are provided to keep the tape moving from end to end zone.

MAINDEC-08-D3EB-D

TC01 Extended Memory Exerciser

TC01 Extended Memory Exerciser is a test program for the PDP-8/E which tests the transfer of data between the TC01 DECtape Control and extended memory fields (more than 4K). It does this by storing a data pattern in an extended memory field, transferring the data onto DECtape, and then reading the data back into the field and checking it for correct transfer.

DEC-08-D3FC-D

Incremental Tape Delay Test

This program is written for incremental tape units interfaced with a Family-of-8 computer. It exercises the tape units and performs a delay test.

MAINDEC-08-D3RA-D

DECTREX 1

TC01 Random Exerciser

DECTREX 1 is a DECtape Random Exerciser for the TC01 DECtape control and any configuration of one to eight TU55 DECtape transports. Drive selection, tape direction, number of blocks, sequence of operation, and patterns generated are by random selection. The DECtape functions exercised are search, read data and write data in normal and continuous modes, read all in continuous mode, and move.

Also included are a short series of processor tests that are executed while waiting for interrupts and during data breaks while searching, reading, and writing from DECtape.

Memory Parity Checkerboard

The PDP-8/E Memory Parity Checkerboard diagnostic tests the parity bit plane for core failure on half-selected lines under worst-case conditions. Its use is intended for basic 4K memory systems.

Extended Memory Parity Test

The PDP-8/E Extended Memory Parity Test is designed to provide worst-case half-select noise conditions within the parity bit plane. Four data patterns and their complements are written and checked for parity errors after each pattern is written. The program tests systems equipped with 8K words of core memory.

Operation of the program is similar to the Extended Memory Checkerboard test, except that program relocation is not included and error halts are provided for error identification.

MAINDEC-08-D4CB-D

PDP-8/E Incremental

Tape Compatibility Test

The PDP-8/E Incremental Tape Compatibility Test verifies that the TU22, 25, 28 Incremental Write-Synchronous Read Transports generate IBM-compatible format. This is accomplished by generating a test tape using pattern #1 of the TC59 Data Reliability Test. The TC59 is the control unit for TU20 transport using the appropriate TC59 Data Reliability Test program. For operating procedures of TC59 Data Reliability Test (7 or 9 track), refer to MAINDEC-9A-D4DC-D and MAINDEC-9A-D4EB-D.

MAINDEC-08-D4FA-D

Incremental Random Exerciser

This program was written for write/read incremental tape units, and exercises several tape units with separate routines. Data errors from the reading of the tape are indicated on TTY printouts.

MAINDEC-08-D4EB-D

TR22, 25, 28 Incremental

Tape Instruction Test

The Incremental Tape Instruction Test is designed to test all IOT instructions associated with the TR8-E. The program may be used on R/W transports, and 7- and 9-track transports at any of the three standard

bit densities. The program exercises each IOT, but does not test validity of the LRCC or CRC. These characters are checked in the data reliability diagnostic.

The program initially tests as many functions as possible without a supply reel mounted. A supply reel is then mounted, and all remaining tape movement IOTs are checked. Instructions which the operator must follow are printed on the TTY.

MAINDEC-08-D5CD-D

DF32 Disk Data

MiniDisk, Interface, Address, Data Test

The DF32 Disk Data is a complete test of the disk system. Also included is a short processor test that is executed while waiting for interrupts, and during data breaks.

MAINDEC-08-D5DB-D

Exerciser for Master and Slaves Units

"Multi Disk" is a high-speed exerciser intended for multi-disk configuration of DF32 disks and can control one to four disks.

MAINDEC-08-D5EB-D

RF08 Disk Data

This program is a complete test of the RF08 disk system. It exercises both read and write functions and checks for errors.

MAINDEC-08-D5FA-D

Multidisk

This is a high-speed exercise program for a multi-disk configuration of RF08 disks.

MAINDEC-08-D6GC-D

A/D Calibration Check

The A/D Calibration Check for the converters is used to ascertain the accuracy of converter adjustments. This tape is used with an adjustable voltage source. The converted value is displayed in the AC, and the switch register is used to select multiplexer channels. Passing of these checks does not guarantee 100% monotonicity and steady state accuracy, since all of the 4096 possible conditions are not checked.

MAINDEC-08-D6HA-D

AF04A Diagnostic and Demonstration

The Diagnostic and Demonstration program for the AF04A allows the operator to type in up to 1000$_8$ pseudo instructions and causes analog to digital conversions via the AF04A. The pseudo-instructions which make up the individual pseudo-program are executed when a "$" (dollar sign) is input from the keyboard. The operator may specify all parameters of the conversion instruction and specify any order of instructions.

MAINDEC-08-D6JD-D

AD08 Diagnostic

For the AD08A this is an I/O Instruction and calibration check. For the AD08B it is also a limited test of multiplexer selection and A/D repeatability.

MAINDEC-08-D6TA-D

AA05/AA07 Calibration Tape

These routines are used for the calibration of the AA05 Digital to Analog Converter and the AA07 Expander unit.

MAINDEC-08-D9AC-D

TC58 Data Reliability Test (7 Track)

The TC58 Data Reliability Test is primarily designed for the collection of statistical information pertaining to the data reliability of the tape drives that may be associated with the TC58 Magnetic Tape Control. The program is also designed to be usable as an aid to the hardware debugging and maintenance of the TC58 Magnetic Tape Control and its associated magnetic tape drives. This program may also be used as an extended data reliability acceptance test.

MAINDEC-08-D9BA-D

TC58 Drive Function Timer

The TC58 Drive Function Timer program is designed to be an aid in the hardware debugging and maintenance of the TC58 Magnetic Tape Control and its associated magnetic tape drives. The program operates on any configuration of 1 to 8, 45 or 75 inch-per-second, 7- or 9-track drives. Selected operations are initiated and timed, and the times are then typed in decimal milliseconds. There is no limit checking on times by the program; the decisions on the validity of times typed must be made external to the program or by the person operating this test.

MAINDEC-08-D9CB-D

TC58 Random Exerciser

The TC58 Random Exerciser Test is a test program designed to simulate tape system usage. Any configuration of 1 through 8 TU20 (or similar) 7- and/or 9-track drives may be concurrently tested.

MAINDEC-08-D9DD-D

TC58 Instruction Test—Part 1

The TC58 Instruction Test is a series of incremental subtests designed to aid in the checkout and maintenance of the TC58 magnetic tape system.

MAINDEC-08-D9EC-D

TC58 Instruction Test—Part 2

The TC58 Instruction Test is a series of incremental subtests designed to aid in the checkout and maintenance of the TC58 magnetic tape system.

MAINDEC-08-D9FC-D

TC58 Data Reliability Test (9 Track)

The TC58 Data Reliability Test is designed primarily for the collection of statistical information pertaining to the data reliability of the tape drives that may be associated with the TC58 Magnetic Tape Control. The program is also designed to be usable as an aid to the hardware debugging and maintenance of the TC58 Magnetic Tape Control and its associated magnetic tape drives. This program may also be used as an extended data reliability acceptance test.

MAINDEC-8I-D01C-D

Instruction Test 1
This is a diagnostic program for testing the AND, TAD, and OPERATE instructions of the PDP-8/E.

MAINDEC-8I-D02B-D

Instruction Test 2
This program is an extensive test of autoindexing, indirect addressing, and the DCA instruction for the PDP-8/E. It also offers minimal testing for interrupt and the AND, TAD, ISZ, JMP, and JMS instructions.

MAINDEC-8I-D5BB-D

DF32 Diskless Logic Test, MiniDisk
Diskless is a test of the DF32 disk logic and its computer interface. This program does not test the disk, nor associated analog interface circuits. The disk is not needed for these routines; if the disk is connected, the disk motor should be turned off. For a complete test of the disk system, use DF32 Disk Data Test.

MAINDEC-8I-D6AC-D

AX08 Diagnostic
This unit is tested in three sections; an instruction test of the logic, a display test for the scope, and a calibration section for the A/D Converter.

MAINDEC-8I-D6CE-D

KV8/I Display Diagnostic
This program uses display routines to exercise the CRT and show its proper operation. Improper operation is indicated by diagnostic routines.

MAINDEC-T8-D8AA-D

Time Sharing-8 Option Test
This is a test of the time sharing option, and is based upon switching between the executive and user modes, testing all "trappable" instructions.

MAINDEC-T8-D8BB-D

TIME-SHARE-8 Hardware Exerciser
TIME-SHARE-8 Hardware Exerciser is a program written to ensure that the hardware can function properly both in and out of time-share mode with data break devices running. It does this by exercising several data break devices simultaneously (if available) and running a program in time-share mode.

APPENDIX B

TABLE OF INSTRUCTIONS

TABLE	INSTRUCTIONS
B-1	PDP-8/E Memory Reference Instructions
B-2	Loading Constants into AC
B-3	Group 1 Operate Microinstructions
B-4	Group 2 Operate Microinstructions
B-5	Group 3 Operate Microinstructions
B-6	Programmed Data Transfer Instructions
B-7	KE8-E Extended Arithmetic Element
B-8	KM8-E Memory Extension
B-9	KM8-E Time Share
B-10	MP8-E Memory Parity
B-11	DK8-EA Line Frequency Clock
B-12	DK8-EC Crystal Clock
B-13	DK8-EP Programmable Real Time Clock
B-14	KP8-E Power Fail Detect
B-15	Teletype Keyboard Reader
B-16	Teletype Teleprinter Punch
B-17	PR8-E Paper Tape Reader
B-18	PP8-E Paper Tape Punch
B-19	PC8-E Reader/Punch
B-20	VC8-E Oscilloscope Display Control
B-21	VS8-E Display Control
B-22	XY8-E Incremental Plotter Control
B-23	LE-8 Line Printer
B-24	DP8-EA/EB Synchronous Modem Interface
B-25	DP8-EP Redundancy Check Option
B-26	CR8-E Card Reader & Control
B-27	AD8-EA A/D Converter
B-28	AM8-EA 8-Channel Analog Multiplexer
B-29	DR8-EA 12-Channel Buffered Digital I/O
B-30	BB08-P General Purpose Interface Unit
B-31	RK08-P Control & RK01 Disk Drive & Control
B-32	DF32-D Disk File & Control
B-33	TC08-P DECtape Control
B-34	AD01A 12-bit Analog-to-Digital Converter
B-35	AFC8 Low-level Analog Multiplexer
B-36	AF04A Guarded Scanning Integrated Digital Voltmeter (IDVM)
B-37	AA50 Digital-to-Analog Converter
B-38	AA05A/AA07 Digital-to-Analog Converter and Control
B-39	Universal Digital Controller (UDC)
B-40	Multicycle Data Break Locations

Unless otherwise noted, all instructional times are 1.2 μs

Table B-1 PDP-8/E Memory Reference Instruction (Refer to Chapter 3)

Mnemonic Symbol	Octal Code	Indicators	Execution Times: Direct Address	Indirect Address	Auto-indexed	Operation
AND Y	0	IR = 0,F,E	2.6	3.8	4.0	Logical AND between Y and AC
TAD Y	1	IR = 1,F,E	2.6	3.8	4.0	Two's complement Add Y to AC
ISZ Y	2	IR = 2,F,E	2.6	3.8	4.0	Increment Y and skip if zero
DCA Y	3	IR = 3,F,E	2.6	3.8	4.0	Deposit at Y and clear AC
JMS Y	4	IR = 4,F,E	2.6	3.8	4.0	Jump to subroutine at Y
JMP Y	5	IR = 5,F	1.2	2.4	2.6	Jump to Y

Table B-2 Loading Constants Into The AC

Numerical Load Definitions for the PDP-8/E Computer are shown below. The six-character symbols in the mnemonic column may be placed in a program and used to set the contents of the Accumulator to the octal value equal to the four numerals in the mnemonic. The symbolic code being assembled must include the mnemonics and the octal code to be substituted for them; each pair is combined by an equal (=) sign. The value which is substituted by the assembler is a microprogrammed operate instruction that performs the sequence of operations listed in the third column.

Mnemonic	Decimal Constant	Octal Code	Instructions Combined
NL0000 =	0	7300	CLA CLL
NL0001 =	1	7301	CLA CLL IAC
NL0002 =	2	7305	CLA CLL IAC RAL
		(or)	
NL0002 =	2	7326	CLA CLL CML RTL
NL0003 =	3	7325	CLA CLL CML IAC RAL
NL0004 =	4	7307	CLA CLL IAC RTL
NL0006 =	6	7327	CLA CLL CML IAC RTL
NL0100 =	64	7203	CLA IAC BSW
NL2000 =	1024	7332	CLA CLL CML RTR
NL3777 =	2047	7350	CLA CLL CMA RAR

Mnemonic	Decimal Constant	Octal Code	Instructions Combined
NL4000 =	−0	7330	CLA CLL CML RAR
NL5777 =	−1025	7352	CLA CLL CMA RTR
NL6000 =	−1024	7333	CLA CLL CML IAC RTL
NL7775 =	−3	7346	CLA CLL CMA RTL
NL7776 =	−2	7344	CLA CLL CMA RAL
NL7777 =	−1	7340	CLA CLL CMA

Table B-3 Group 1 Operate Microinstructions

Mnemonic Symbol	Octal Code	Sequence	Operation
NOP	7000	—	No operation. Causes a 1.2 μs program delay.
IAC	7001	3	Increment AC. The content of the AC is incremented by one in two's complement arithmetic.
RAL	7004	4	Rotate AC and L left. The content of the AC and the L are rotated left one place.
RTL	7006	4	Rotate two places to the left. Equivalent to two successive RAL operations.
RAR	7010	4	Rotate AC and L right. The content of the AC and L are rotated right one place.
RTR	7012	4	Rotate two places to the right. Equivalent to two successive RAR operations.
BSW	7002	4	Byte swap.
CML	7020	2	Complement L.
CMA	7040	2	Complement AC. The content of the AC is set to the one's complement of its current content.
CIA	7041	2, 3	Complement and increment accumulator. Used to form two's complement.
CLL	7100	1	Clear L.
CLL RAL	7104	1, 4	Shift positive number one left.
CLL RTL	7106	1, 4	Clear link, rotate two left.
CLL RAR	7110	1, 4	Shift positive number one right.
CLL RTR	7112	1, 4	Clear link, rotate two right.

Table B-3 (Cont.) Group 1 Operate Microinstructions

Mnemonic Symbol	Octal Code	Sequence	Operation
STL	7120	1, 2	Set link. The L is set to contain a binary 1.
CLA	7200	1	Clear AC. To be used alone or in OPR 1 combinations.
CLA IAC	7201	1, 3	Set AC = 1.
GLK	7204	1, 4	Get link. Transfer L into AC11.
CLA CLL	7300	1	Clear AC and L.
STA	7240	2	Set AC = 1. Each bit of the AC is set to contain a 1.

Table B-4 Group 2 Operate Microinstructions

Mnemonic Symbol	Octal Code	Sequence	Operation
HLT	7402	3	Halt. Stops the program after completion of the cycle in process. If this instruction is combined with others in the OPR 2 group the other operations are completed before the end of the cycle.
OSR	7404	3	OR with switch register. The OR function is performed between the content of the SR and the content of the AC, with the result left in the AC.
SKP	7410	1	Skip, unconditional. The next instruction is skipped.
SNL	7420	1	Skip if L $\neq$ 0.
SZL	7430	1	Skip if L = 0.
SZA	7440	1	Skip if AC = 0.
SNA	7450	1	Skip if AC $\neq$ 0.
SZA SNL	7460	1	Skip if AC = 0, or L $\neq$ 1, or both.
SNA SZL	7470	1	Skip if AC $\neq$ 0 and L = 0.
SMA	7500	1	Skip on minus AC. If the content of the AC is a negative number, the next instruction is skipped.
SPA	7510	1	Skip on positive AC. If the content of the AC is a positive number, including zero, the next instruction is skipped.

Table B-4 (Cont.) Group 2 Operate Microinstructions

Mnemonic Symbol	Octal Code	Sequence	Operation
SMA SNL	7520	1	Skip if AC $<$ 0, or L $=$ 1, or both.
SPA SZL	7530	1	Skip if AC $\geqslant$ 0 and if L $=$ 0.
SMA SZA	7540	1	Skip if AC $\leqslant$ 0.
SPA SNA	7550	1	Skip if AC $>$ 0.
CLA	7600	2	Clear AC. To be used alone or in OPR 2 combinations.
LAS	7604	1, 3	Load AC with SR
SZA CLA	7640	1, 2	Skip if AC $=$ 0, then clear AC.
SNA CLA	7650	1, 2	Skip if AC $\neq$ 0, then clear AC.
SMA CLA	7700	1, 2	Skip if AC $<$ 0, then clear AC.
SPA CLA	7710	1, 2	Skip if AC $\geqslant$ 0, then clear AC.

Table B-5 Group 3 Operate Microinstructions

Mnemonic Symbol	Octal Code	Operation
NOP	7401	No Operation
MQL	7421	Load Multiplier Quotient
MQA	7501	Multiplier Quotient OR into Accumulator
SWP	7521	Swap Accumulator and Multiplier Quotient
CLA	7601	Clear Accumulator
CLA, MQL	7621	Clear Accumulator and Multiplier Quotient
CLA, MQA	7701	Clear Accumulator, Load Multiplier Quotient into Accumulator
CLA, SWP	7721	Load Multiplier Quotient into Accumulator, Clear Multiplier Quotient

Table B-6 Programmed Data Transfer Instructions

Mnemonic Symbol	Octal Code	Operation
ION	6001	Interrupt Turn On
IOF	6002	Interrupt Turn Off
SKON	6000	Skip if Interrupt On, IOF
SRQ	6003	Skip if Interrupt Request
GTF	6004	Get Flags
RTF	6005	Restore Flag, ION
SGT	6006	Skip if "Greater Than" Flag is Set
CAF	6007	Clear All Flags

Table B-7 KE8-E Extended Arithmetic Element

Mnemonic Symbol	Octal Code	Operation	Time (μs)
Standard			
NOP	7401	No Operation	
MQL	7421	Load Multiplier Quotient	
MQA	7501	Multiplier Quotient OR Load into Accumulator	
SWP	7521	Swap Accumulator and Multiplier Quotient	
CLA	7601	Clear Accumulator	
CLA, MQL	7621	Clear Accumulator and Multiplier Quotient	
CLA, MQA	7701	Clear Accumulator, Load Multiplier Quotient into Accumulator	
CLA, SWP	7721	Load Multiplier Quotient into Accumulator, Clear Multiplier Quotient	
Optional			
SGT	6006	Skip if Greater Than	
ACS	7403	Store Accumulator in Step Counter	
MUY	7405	Multiply	7.4
DVI	7407	Divide	7.7
NMI	7411	Normalize	1.2 + .3N
SHL	7413	Shift Left	2.6 + .3N
ASR	7415	Arithmetic Shift Right	2.6 + .3N
LSR	7417	Logical Shift Right	2.6 + .3N
SCA	7441	OR Step Counter into Accumulator	
NMU	7451	Absolute Normalize	1.2 + .3N
SAM	7453	MQ minus AC to AC	
DPSZ	7457	Double Precision Skip if Zero	
DAD	7563	Double Precision Add	5.2
DST	7565	Double Precision Store	5.2
DCM	7567	Double Precision Two's Complement	1.8
DPIC	7575	Double Precision Increment	2.0

Table B-8 KM8-E Memory Extension

Mnemonic Symbol	Octal Code	Operation
GTF	6004	Get Flags
RTF	6005	Restore Flags, ION
CDF	62N1	Change to Data Field N (N=0 to 7)
CIF	62N2	Change to Instruction Field N (N=0 to 7)
CDF, CIF	62N3	Change Data Field, Change Instruction Field
RDF	6214	Read Data Field
RIF	6224	Read Instruction Field
RIB	6234	Read Interrupt Buffer
RMF	6244	Restore Memory Field

Table B-9 KM8-E Time-Share

Mnemonic Symbol	Octal Code	Operation
CINT	6204	Clear User Interrupt
SINT	6254	Skip on User Interrupt
CUF	6264	Clear User Flag
SUF	6274	Set User Flag

Table B-10 MP8-E Memory Parity

Mnemonic Symbol	Octal Code	Operation
DPI	6100	Disable Memory Parity Error Interrupt
SMP	6101	Skip on No Memory Parity Error
EPI	6103	Enable Memory Parity Error Interrupt
CMP	6104	Clear Memory Parity Error Flag
SMP, CMP	6105	Skip on No Memory Parity Error, Clear Memory Parity Error Flag
CEP	6106	Check for Even Parity
SPO	6107	Skip on Memory Parity Option

Table B-11 DK8-EA Line Frequency Clock

Mnemonic Symbol	Octal Code	Operation
CLEI	6131	Enable Interrupt
CLDI	6132	Disable Interrupt
CLSK	6133	Skip on Clock Flag and Clear Flag

Table B-12 DK8-EC Crystal Clock

Mnemonic Symbol	Octal Code	Operation
CLEI	6131	Enable Interrupt
CLDI	6132	Disable Interrupt
CLSK	6133	Skip on Clock Flag and Clear Flag

Table B-13 DK8-EP Programmable Real Time Clock

Mnemonic Symbol	Octal Code	Operation
CLZE	6130	Clear Clock Enable Register per AC
CLSK	6131	Skip on Clock Interrupt
CLOE	6132	Set Clock Enable Register per AC
CLAB	6133	AC to Clock Buffer
CLEN	6134	Load Clock Enable Register
CLSA	6135	Clock Status to AC
CLBA	6136	Clock Buffer to AC
CLCA	6137	Clock Counter to AC

Table B-14 KP8-E Power Fail Detect

Mnemonic Symbol	Octal Code	Operation
SPL	6102	Skip on Power Low

Table B-15 Teletype Keyboard Reader

Mnemonic Symbol	Octal Code	Operation
KCF	6030	Clear Keyboard Flag
KSF	6031	Skip on Keyboard Flag
KCC	6032	Clear Keyboard Flag, and AC, Advance Reader
KRS	6034	Read Keyboard Buffer Static
KIE	6035	Set/Clear Interrupt Enable
KRB	6036	Read Keyboard Buffer, Clear Flag

Table B-16 Teletype Teleprinter/Punch

Mnemonic Symbol	Octal Code	Operation
TFL	6040	Set Teleprinter Flag
TSF	6041	Skip on Teleprinter Flag
TCF	6042	Clear Teleprinter Flag
TPC	6044	Load Teleprinter and Print
TSK	6045	Skip on Printer or Keyboard Flag
TLS	6046	Load Teleprinter Sequence

Table B-17 PR8-E, PR8-ES Paper Tape Readers

Mnemonic Symbol	Octal Code	Operation
RPE	6010	Set Reader/Punch Interrupt Enable
RSF	6011	Skip on Reader Flag
RRB	6012	Read Reader Buffer
RFC	6014	Reader Fetch Character
RRB, RFC	6016	Read Buffer and Fetch New Character
PCE	6020	Clear Reader/Punch Interrupt Enable

Table B-18 PP8-E Paper Tape Punch

Mnemonic Symbol	Octal Code	Operation
RPE	6010	Set Reader/Punch Interrupt Enable
PCE	6020	Clear Reader/Punch Interrupt Enable
PSF	6021	Skip on Punch Flag
RCF	6022	Clear Punch Flag
PPC	6024	Load Punch Buffer and Punch Character
PLS	6026	Load Punch Buffer Sequence

Table B19 PC8-E Reader/Punch

Mnemonic Symbol	Octal Code	Operation
RPE	6010	Set Reader/Punch Interrupt Enable
RSF	6011	Skip on Reader Flag
RRB	6012	Read Reader Buffer
RFC	6014	Reader Fetch Character
RFC, RRB	6016	Read Buffer and Fetch New Character
PCE	6020	Clear Reader/Punch Interrupt Enable
PSF	6021	Skip On Punch Flag
PCF	6022	Clear Punch Flag
PPC	6024	Load Punch Buffer and Punch Character
PLS	6026	Load Punch Buffer Sequence

Table B-20 VC8-E Oscilloscope Display Control

Mnemonic Symbol	Octal Code	Operation
DIEN	6050	Load Interrupt Enable Register from Accumulator
DSPD	6051	Skip on Point Done Flag and Clear Point Done Flag

Table B-20 (Cont.) VC8-E Oscilloscope Display Control

Mnemonic Symbol	Octal Code	Operation
DSCD	6052	Skip on Character Done Flag, Clear Character Done Flag
DSLP	6053	Skip on Light Pen Flag, Clear Light Pen Flag
DSDF	6054	Skip on Any Display Flag
DRIS	6055	Read Interrupt Flag into Accumulator, Clear Flags
DSFF	6056	Set Fast Plotting Mode
DCFF	6057	Clear Fast Plotting Mode
DLXA	6060	Load X Register From Accumulator
DLXB	6061	Load X Register from Accumulator, Clear Accumulator
DLXC	6062	Load X Register from Accumulator and Intensify
DLXD	6063	Load X Register from Accumulator, Clear Accumulator, and Intensify
DSIA	6064	Set Small Character Size
DSIB	6065	Set Large Character Size
DINX	6066	Increment X and Intensify
DSC	6067	Display Character from Acumulator
DLYA	6070	Load Y Register from Accumulator
DLYB	6071	Load Y Register from Accumulator, Clear Accumulator
DLYC	6072	Load Y Register from Accumulator, and Intensify
DLYD	6073	Load Y Register from Accumulator, Clear Accumulator, and Intensify
DCH1	6074	Set VR12 Channel 1
DCH2	6075	Set VR12 Channel 2
DINY	6076	Increment Y and Intensify
DSCI	6077	Clear Status and Interrupt Enable Registers

Table B-21 VS8-E Display Control

Mnemonic Symbol	Octal Code	Operation
SNC	6051	Skip if Cursor Interrupt Not Set
CFF	6052	Clear Interrupt Flag
SAC	6062	Select Analog Comparator
LDF	6063	Load and Select Format
LDX	6064	Load Accumulator into X Sample and Hold
LDY	6065	Load Accumulator into Y Sample and Hold
EXC	6066	Execute Microprogrammed Instruction
EXL	6067	Execute Long Vector Microprogrammed Instruction

Table B-21 (Cont.) VS8-E Display Control

Mnemonic Symbol	Octal Code	Operation
SRF	6071	Skip if Ready Flag Is Set
CRF	6072	Clear Ready Flag
SDA	6073	Skip if D/A Greater than Analog Comparison Signal
LDA	6074	Load D/A

Table B-22 XY8-E Incremental Plotter Control

Mnemonic Symbol	Octal Code	Operation
PLCE	6500	Clear Interrupt Enable
PLSF	6501	Skip on Plotter Flag
PLCF	6502	Clear Plotter Flag
PLPU	6503	Pen Up
PLLR	6504	Load Direction Register, Set Flag
PLPD	6505	Pen Down
PLCF, PLLR	6506	Clear Flag, Load Direction Register Set Flag
PLSE	6507	Set Interrupt Enable

Table B-23 LE-8 Line Printer

Mnemonic Symbol	Octal Code	Operation
PSKF	6661	Skip on Character Flag
PCLF	6662	Clear the Character Flag
PSKE	6663	Skip on Error
PSTB	6664	Load Printer Buffer, Print on Full Buffer or Control Character
PSIE	6665	Set Program Interrupt Flag
PCLF, PSTB	6666	Clear Line Printer Flag, Load Character, and Print
PCIE	6667	Clear Program Interrupt Flag

Table B-24 Synchronous Modem Interface

Mnemonic Symbol	Octal Code	Operation
SGTT	6405	Transmit Go
SGRR	6404	Receive Go
SSCD	6400	Skip if Character Detected

Table B-24 (Cont.) Synchronous Modem Interface

Mnemonic Symbol	Octal Code	Operation
SCSD	6406	Clear Sync Detect
SSRO	6402	Skip if Receive Word Count Overflow
SCSI	6401	Clear Synchronous Interface
SRTA	6407	Read Transfer Address Register
SLCC	6412	Load Control
SSRG	6410	Skip if Ring Flag
SSCA	6411	Skip if Carrier/AGC Flag
SRS2	6414	Read Status 2
SRS1	6415	Read Status 1
SLFL	6413	Load Field
SSBE	6416	Skip on Bus Error
SRCD	6417	Read Character Detected (if AC0=0) Maintenance Instruction (if AC0=1)
SSTO	6403	Skip if Transmit Word Count Overflows

Break Address Locations

7720	
7721	Test Characters
7722	
7723	
7724	Receive Word Count
7725	Receive Current Address
7726	Not Used
7727	Transmit Word Count
7730	Transmit Current Address

For additional interfaces

Device Codes	Break Locations
42, 43	7700-7710
44, 45	7660-7670
46, 47	7640-7650

Table B-25 DP8-EP Redundancy Check Option

Mnemonic Symbol	Octal Code	Operation
RCTV	6110	Test VRC and Skip
RCRL	6111	Read BCC Low
RCRH	6112	Read BCC High
RCCV	6113	Compute VRC
RCGB	6114	Generate BCC
RCLC	6115	Load Control
RCCB	6116	Clear BCC Accumulation

Table B-26 CR8-E Card Reader and Control or CM8-E Optical Mark Card Reader and Control

Mnemonic Symbol	Octal Code	Operation
RCSF	6631	Skip on Data Ready
RCRA	6632	Read Alphanumeric
RCRB	6634	Read Binary
RCNO	6635	Read Conditions Out to Card Reader
RCRC	6636	Read Compressed
RCNI	6637	Read Condition In from Card Reader
RCSD	6671	Skip on Card Done Flag
RCSE	6672	Select Card Reader and Skip if Ready
RCRD	6674	Clear Card Done Flag
RCSI	6675	Skip If Interrupt Being Generated
RCTF	6677	Clear Transition Flags

Table B-27 AD8-EA A/D Converter

Mnemonic Symbol	Octal Code	Operation
ADEI	6530	Set A/D Converter Interrupt Enable
ADSF	6531	Skip on A/D Done Flag
ADCL	6532	Clear A/D Done and Timing Flip-Flop
ADSK	6533	Skip on A/D Timing Error
ADRB	6534	Read A/D Converter Buffer
ADCV	6535	Convert Analog Voltage to Digital Value
ADST	6536	Read A/D Buffer and Start Conversion
ADDI	6537	Clear Interrupt Enable Flip-Flop

Table B-28 AM8-EA 8-Channel Analog Multiplexer

Mnemonic Symbol	Octal Code	Operation
AMLC	6541	Load Multiplexer Channel
AMIM	6542	Increment Multiplexer Channel
AMRC	6543	Read Multiplexer Channel
AMSM	6544	Set Multiplexer to Auto-Increment Mode
AMRM	6545	Reset Auto-Increment Mode
AMMS	6546	Load Multiplexer
AMIC	6547	Increment Multiplexer Channel

Table B-29 DR8-EA 12-Channel Buffered Digital I/O

Mnemonic Symbol	Octal Code	Operation
DBJO	6570	Output Register to AC (Jam Transfer)
DBOR	6571	Load Ones in Output Register
DBZR	6572	Load Zeros in Output Register
DBAN	6573	Load Output Register
DBOO	6574	OR Output Register With AC
DBCL	6575	Clear Input Register Stages
DBOI	6576	OR Input Register with AC
DBJI	6577	Input Register to AC (Jam Transfer)

* 6xxl Where xx can be any code (50-57) jumper-selectable on module

Table B-30 BB08-P General Purpose Interface Unit

Mnemonic Symbol	Octal Code	Operation	Time (μs)
GTSF	6361	Skip on Transmit Flag	2.6
GCTF	6362	Clear Transmit Flag	2.6
	6564	(User-Assigned)	2.6
GRSF	6371	Skip on Receive Flag	2.6
GCRF	6372	Clear Receive Flag	2.6
GRDB	6374	Read Device Buffer	2.6

Table B-31 RK08-P Control and RK01 Disk Drive and Control

Mnemonic Symbol	Octal Code	Operation	Time (μs)
DLDA	6731	Load Disk Address (Maintenance Only)	2.6
DLDC	6732	Load Command Register	2.6
DLDR	6733	Load Disk Address and Read	3.6
DRDA	6734	Read Disk Address	2.6
DLDW	6735	Load Disk Address and Write	2.6
DRDC	6736	Read Disk Command Register	3.6
DCHP	6737	Load Disk Address and Check Parity	4.6
DRDS	6741	Read Disk Status Register	2.6
DCLS	6742	Clear Status Register	2.6
DMNT	6743	Load Maintenance Register	3.6
DSKD	6745	Skip on Disk Done	3.6
DSKE	6747	Skip on Disk Error	4.6
DCLA	6751	Clear All	2.6
DRWC	6752	Read Word Count Register	2.6
DLWC	6753	Load Word Count Register	3.6
DLCA	6755	Load Current Address Register	3.6
DRCA	6757	Read Current Address Register	4.6

Table B-32 DF32-D Disk File and Control

Mnemonic Symbol	Octal Code	Operation	Time (μs)
DCMA	6601	Clear Disk Address Register	2.6
DMAR	6603	Load Disk Address Register and Read	3.6
DMAW	6605	Load Disk Address Register and Write	3.6
DCEA	6611	Clear Disk Extended Address	2.6
DSAC	6612	Skip on Address Confirmed Flag	2.6
DEAL	6615	Load Disk Extended Address	3.6
DEAC	6616	Read Disk Extended Address	3.6
DFSE	6621	Skip on Zero Error Flag	2.6
DFSC	6622	Skip on Data Completion Flag	2.6
DMAC	6626	Read Disk Memory Address Register	3.6

Address Locations:

7750 = Word Count
7751 = Memory Address

Table B-33 TC08-P DECtape Control

Mnemonic Symbol	Octal Code	Operation	Time (μs)
DTRA	6761	Read Status Register A	2.6
DTCA	6762	Clear Status Register A	2.6
DTXA	6764	Load Status Register A	2.6
DTLA	6766	Clear and Load Status Register A	3.6
DTSF	6771	Skip on Flag	2.6
DTRB	6772	Read Status Register B	2.6
DTXB	6774	Load Status Register B	2.6

Address Locations:

7754 = Word Count
7755 = Current Address

Table B-34 AD01A 12-Bit Analog-to-Digital Converter

Mnemonic Symbol	Octal Code	Operation	Time (μs)
ADSF	6531	Skip on A/D Done Flag	2.6
ADRB	6532	Read A/D Buffer	2.6
ADCV	6534	Convert Analog Input	2.6
ADSC	6535	Select Multiplexer Channel and Gain	3.6
ADRC	6536	Read A/D Buffer, Clear Flag, and Start Conversion	3.6
ADSR	6537	Select Channel and Gain and Read A/D Buffer	4.6

Table B-35 AFC-8 Low-Level Analog Multiplexer

Mnemonic Symbol	Octal Code	Operation	Time (μs)
ADSG	6542	Set Multiplexer Gain	2.6
ADSA	6544	Set Multiplexer Address	2.6
ADSF	6531	Skip on A/D Flag	2.6
ADRB	6534	Read A/D Converter Buffer	2.6
ADSG	6542	Set Multiplexer Gain	2.6
ADSA	6544	Set Multiplexer Address	2.6

Table B-36 AF04A Guarded Scanning Integrated Digital Voltmeter (IDVM)

Mnemonic Symbol	Octal Code	Operation	Time (μs)
VCNV	6541	Select Channel and Convert	2.6
VSEL	6542	Select Range and Gate	2.6
VINX	6544	Index Channel and Convert	2.6
VSDR	6561	Skip on Data Ready	2.6
VRD	6562	Read Data and Clear Flag	2.6
VBA	6564	Byte Advance	2.6
VSCC	6571	Sample Current Channel	2.6

Table B-37 AA50 Digital-to-Analog Converter

Mnemonic Symbol	Octal Code	Operation	Time (μs)
DAC 1	6551	Select DAC 1	2.6
DAC 2	6552	Select DAC 2	2.6
DAC 3	6553	Select DAC 3	3.6
DAC 4	6554	Select DAC 4	2.6
DAC 5	6555	Select DAC 5	3.6
DAC 6	6556	Select DAC 6	3.6

Table B-38 AA05A/AA07 Digital-to-Analog Converter and Control

Mnemonic Symbol	Octal Code	Operation	Time (μs)
DACL	6551	Clear DAC Address	2.6
DALD	6552	Load DAC Address	2.6
DALI	6562	Load DAC Input Register	2.6
DAUP	6564	Update All Channels	2.6

Table B-39 Universal Digit Controller (UDC)

Mnemonic Symbol	Octal Code	Operation	Time (μs)
UDSS	6351	Skip on Scan Not Busy	2.6
UDSC	6353	Start Interrupt Scan	3.6
UDRA	6356	Read Address and Generic Type	3.6
UDLS	6357	Load Previous Status	4.6
UDSF	6361	Skip on UDC Flag and Clear Flag	2.6
UDLA	6363	Load Address	3.6
UDEI	6364	Enable UDC Interrupt Flag	2.6
UDDI	6365	Disable UDC Interrupt Flag	3.6
UDRD	6366	Clear AC and Read Data	3.6
UDLD	6367	Load Data and Clear AC	4.6

Table B-40 Multicycle Data Break Locations

Assigned Locations	Data Break Device	Channel
7640-7650	DP8-EA/EB	4
7660-7670	DP8-EA/EB	3
7700-7710	DP8-EA/EB	2
7720-7730	DP8-EA/EB	1
7750,7751	DF32-D	
7752,7753	(Reserved for Industry Standard Magnetic Tape)	
7754,7755	TC08-P	

APPENDIX C

TABLES OF CODES

TABLE	TITLE
C-1	Model 33 ASR/KSR Teletype Code (ASCII) in Octal Form
C-2	Model 33 ASR/KSR Teletype Code (ASCII) in Binary Form
C-3	Card Reader Code

Table C-1 Model 33 ASR/KSR Teletype Code (ASCII) in Octal Form

Character	8-Bit Code (in octal)	Character	8-Bit Code (in octal)	Character	8-Bit Code (in octal)
A	301	1	261	=	275
B	302	2	262	>	276
C	303	3	263	?	277
D	304	4	264	@	300
E	305	5	265	[	333
F	306	6	266	\	334
G	307	7	267	]	335
H	310	8	270	↑	336
I	311	9	271	←	337
J	312				
K	313	!	241	Leader/Trailer	200
L	314	"	242	Line-Feed	212
M	315	#	243	Carriage-Return	215
N	316	$	244	Space	240
O	317	%	245	Rub-out	377
P	320	&	246	Blank	000
Q	321	'	247	Alt-mode	375
R	322	(	250		
S	323	)	251	WRU	205
T	324	*	252	TAPE	222
U	325	+	253		224
V	326	,	254	TAB	211
W	327	—	255	X OFF	223
X	330	.	256	EOT	204
Y	331	/	257	RU	206
Z	332	:	272	BELL	207
		;	273	VT	213
0	260	<	274	FORM	214

Note 1: Shift key adds hole 5.
Note 2: Control key deletes hole 2.

Table C-2 Model 33 ASR/KSR Teletype Code (ASCII) in Binary Form

1 = HOLE PUNCHED = MARK
0 = NO HOLE PUNCHED = SPACE

MOST SIGNIFICANT BIT
LEAST SIGNIFICANT BIT

				8	7	6	5	4	S	3	2	1
	@	SPACE	NULL/IDLE				0	0		0	0	0
	A	!	START OF MESSAGE				0	0		0	0	1
	B	"	END OF ADDRESS				0	0		0	1	0
	C	#	END OF MESSAGE				0	0		0	1	1
	D	$	END OF TRANSMISSION				0	0		1	0	0
	E	%	WHO ARE YOU				0	0		1	0	1
	F	&	ARE YOU				0	0		1	1	0
	G	'	BELL				0	0		1	1	1
	H	(	FORMAT EFFECTOR				0	1		0	0	0
	I	)	HORIZONTAL TAB				0	1		0	0	1
	J	*	LINE FEED				0	1		0	1	0
	K	+	VERTICAL TAB				0	1		0	1	1
	L	,	FORM FEED				0	1		1	0	0
	M	—	CARRIAGE RETURN				0	1		1	0	1
	N	.	SHIFT OUT				0	1		1	1	0
	O	/	SHIFT IN				0	1		1	1	1
	P	0	DCO				1	0		0	0	0
	Q	1	READER ON				1	0		0	0	1
	R	2	TAPE (AUX ON)				1	0		0	1	0
	S	3	READER OFF				1	0		0	1	1
	T	4	(AUX OFF)				1	0		1	0	0
	U	5	ERROR				1	0		1	0	1
	V	6	SYNCHRONOUS IDLE				1	0		1	1	0
	W	7	LOGICAL END OF MEDIA				1	0		1	1	1
	X	8	S 0				1	1		0	0	0
	Y	9	S 1				1	1		0	0	1
	Z	:	S 2				1	1		0	1	0
	[	;	S 3				1	1		0	1	1
	~	<	S 4				1	1		1	0	0
Alt-mode	]	=	S 5				1	1		1	0	1
	↑	>	S 6				1	1		1	1	0
RUB OUT	←	?	S 7				1	1		1	1	1

1	0	0	SAME
1	0	1	SAME
1	1	0	SAME
1	1	1	SAME

Table C-3 Card Reader Code

INTERNAL CODE	CARD ZONE	CODE NUM.	IBM 26 CHARACTER	IBM 29 CHARACTER
00	NONE		SPACE	SPACE
01	—	1	1	1
02	—	2	2	2
03	—	3	3	3
04	—	4	4	4
05	—	5	5	5
06	—	6	6	6
07	—	7	7	7
10	—	8	8	8
11	—	9	9	9
12	—	8-2	ASSIGNABLE	
13	—	8-3	#	=
14	—	8-4	@	'
15	—	8-5	ASSIGNABLE	
16	—	8-6	ASSIGNABLE	
17	—	8-7	ASSIGNABLE	
20	0	—	0	0
21	0	1	/	/
22	0	2	S	S
23	0	3	T	T
24	0	4	U	U
25	0	5	V	V
26	0	6	W	W
27	0	7	X	X
30	0	8	Y	Y
31	0	9	Z	Z
32	0	8-2	ASSIGNABLE	
33	0	8-3	,	,
34	0	8-4	%	(
35	0	8-5	ASSIGNABLE	
36	0	8-6	ASSIGNABLE	
37	0	8-7	ASSIGNABLE	
40	11	—	—	—
41	11	1	J	J
42	11	2	K	K
43	11	3	L	L
44	11	4	M	M
45	11	5	N	N
46	11	6	O	O
47	11	7	P	P
50	11	8	Q	Q
51	11	9	R	R
52	11	8-2	ASSIGNABLE	
53	11	8-3	$	$
54	11	8-4	*	*
55	11	8-5	ASSIGNABLE	
56	11	8-6	ASSIGNABLE	
57	11	8-7	ASSIGNABLE	
60	12	—	&	+
61	12	1	A	A
62	12	2	B	B
63	12	3	C	C
64	12	4	D	D
65	12	5	E	E
66	12	6	F	F
67	12	7	G	G
70	12	8	H	H
71	12	9	I	I
72	12	8-2	ASSIGNABLE	
73	12	8-3	.	.
74	12	8-4	¤	)
75	12	8-5	ASSIGNABLE	
76	12	8-6	ASSIGNABLE	
77	12	8-7	ASSIGNABLE	

WILL NOT DETECT INVALID PUNCH COMBINATIONS

APPENDIX D
PERFORATED-TAPE LOADER SEQUENCES

READIN MODE LOADER

The readin mode (RIM) loader is a minimum length, basic perforated-tape reader program for the ASR33, it is initially stored in memory by manual use of the operator console keys and switches. The loader is permanently stored in 18 locations of page 37.

A perforated tape to be read by the RIM loader must be in RIM format:

Tape Channel 8 7 6 5 4 S 3 2 1	Format
1 0 0 0 0 . 0 0 0	Leader-trailer code
0 1 A1 . A2 0 0 A3 . A4	Absolute address to contain next 4 digits
0 0 X1 . X2 0 0 X3 . X4	Content of previous 4-digit address
0 1 A1 . A2 0 0 A3 . A4	Address
0 0 X1 . X2 0 0 X3 . X4	Content
(Etc.)	(Etc.)
1 0 0 0 0 . 0 0 0	Leader-trailer code

The RIM loader can only be used in conjunction with the ASR33 reader (not the high-speed perforated-tape reader). Because a tape in RIM format is, in effect, twice as long as it need be, it is suggested that the RIM loader be used only to read the binary loader when using the ASR33. (Note that PDP-8 diagnostic program tapes are in RIM format.)

The complete PDP-8/I RIM loader (SA = 7756) is as follows:

Absolute Address	Octal Content	Tag	Instruction I Z	Comments
7756,	6032	BEG,	KCC	/CLEAR AC AND FLAG
7757,	6031		KSF	/SKIP IF FLAG = 1
7760,	5357		JMP .−1	/LOOKING FOR CHARACTER
7761,	6036		KRB	/READ BUFFER
7762,	7106		CLL RTL	
7763,	7006		RTL	/CHANNEL 8 IN AC0
7764,	7510		SPA	/CHECKING FOR LEADER
7765,	5357		JMP BEG+1	/FOUND LEADER
7766,	7006		RTL	/OK, CHANNEL 7 IN LINK
7767,	6031		KSF	
7770,	5367		JMP .−1	
7771,	6034		KRS	/READ, DO NOT CLEAR
7772,	7420		SNL	/CHECKING FOR ADDRESS
7773,	3776		DCA I TEMP	/STORE CONTENT
7774,	3376		DCA TEMP	/STORE ADDRESS

```
7775,   5356              JMP BEG     /NEXT WORD
7776,   0         TEMP,   0           /TEMP STORAGE
```

Placing the RIM loader in core memory by way of the operator console keys and switches is accomplished as follows:

1. Set the starting address 7756 in the switch register (SR).
2. Press LOAD ADDRESS key.
3. Set the first instruction (6032) in the SR.
4. Press the DEPOSIT key.
5. Set the next instruction (6031) in the SR.
6. Press DEPOSIT key.
7. Repeat steps 5 and 6 until all 16 instructions have been deposited.

To load a tape in RIM format, place the tape in the reader, set the SR to the starting address 7756 of the RIM loader (not of the program being read), press the LOAD ADDRESS key, press the START key, and start the Teletype reader.

Refer to Digital Program Library document DEC-08-LRAA-D for additional information on the Readin Mode Loader program.

BINARY LOADER

The binary loader (BIN) is used to read machine language tapes (in binary format) produced by the program assembly language (PAL). A tape in binary format is about one-half the length of the comparable RIM format tape. It can, therefore, be read about twice as fast as a RIM tape and is, for this reason, the more desirable format to use with the 10 cps ASR33 reader or the Type PR8/I High-Speed Perforated-Tape Reader.

The format of a binary tape is as fol ows:

LEADER: about 2 feet of leader-trailer codes.

BODY: characters representing the absolute, machine language program in easy-to-read binary (or octal) form. The section of tape may contain characters representing instructions (channel 8 and 7 not punched) or origin resettings (channel 8 not punched, channel 7 punched) and is concluded by 2 characters (channel 8 and 7 not punched) that represent a check sum for the entire section.

TRAILER: same as leader.

Example of the format of a binary tape:

Tape Channel 8 7 6 5 4 S 3 2 1	Memory Location	Content	Comments
1 0 0 0 0 . 0 0 0			leader-trailer code
0 1 0 0 0 . 0 1 0 0 0 0 0 0 . 0 0 0		0200	
0 0 1 1 1 . 0 1 0 0 0 0 0 0 . 0 0 0	0200	CLA	origin-setting
0 0 0 0 1 . 0 1 0 0 0 1 1 1 . 1 1 1	0201	TAD 277	
0 0 0 1 1 . 0 1 0 0 0 1 1 1 . 1 1 0	0202	DCA 276	
0 0 1 1 1 . 1 0 0 0 0 0 0 0 . 0 1 0	0203	HLT	
0 1 0 0 0 . 0 1 0 0 0 1 1 1 . 1 1 1		0277	origin-setting
0 0 0 0 0 . 0 0 0 0 0 1 0 1 . 0 1 1	0277	0053	
0 0 0 0 1 . 0 0 0 0 0 0 0 0 . 1 1 1		1007	sum check
1 0 0 0 0 . 0 0 0			leader-trailer code

After a BIN tape has been read in, one of the two following conditions exists:

a. No checksum error: halt with AC = 0
b. Checksum error: halt with AC = (completed checksum) — (tape checksum)

Operation of the BIN loader in no way depends upon or uses the RIM loader. To load a tape in BIN format place the tape in the reader, set the SR to 7777 (the starting address of the BIN loader), press the LOAD ADDRESS key, set SR switch 0 up for loading via the Teletype unit or down for loading via the high speed reader, then press the START key, and start the tape reader.

Refer to Digital Program Library document Digital-8-2-U [DEC-08-LBAA-D] for additional information on the Binary Loader program.

APPENDIX E

LINE PRINTER CHARACTER CODES

LE8-E LINE PRINTER CODE

Octal digits in AC 5-8	Octal digit in AC 9-11							
	0	1	2	3	4	5	6	7
00								
01			LF		FF	CR		
02								
03								
04	SP	!	"	#	$	%	&	'
05	(	)	*	+	,	-	.	/
06	0	1	2	3	4	5	6	7
07	8	9	:	;	<	=	>	?
10	@	A	B	C	D	E	F	G
11	H	I	J	K	L	M	N	O
12	P	Q	R	S	T	U	V	W
13	X	Y	Z	[	\	]	^	←
14	`	a	b	c	d	e	f	g
15	h	i	j	k	l	m	n	o
16	p	q	r	s	t	u	v	w
17	x	y	z	{	|	}	~	■

Notes: 1. LF = Line Feed
FF = Form Feed
CR = Carriage Return
SP = Space

2. Characters below the heavy line are available only on 96-character printers.
3. The dollar sign may be changed to the symbol for pound sterling via a jumper within the printer.
4. All codes for which there is no character (and all codes below the heavy line, in the case of a 64-character printer) are printed as a space.

APPENDIX F

SCALES OF NOTATION

2^x IN DECIMAL

x	2^x	x	2^x	x	2^x
0.001	1.00069 33874 62581	0.01	1.00695 55500 56719	0.1	1.07177 34625 36293
0.002	1.00138 72557 11335	0.02	1.01395 94797 90029	0.2	1.14869 83549 97035
0.003	1.00208 16050 79633	0.03	1.02101 21257 07193	0.3	1.23114 44133 44916
0.004	1.00277 64359 01078	0.04	1.02811 38266 56067	0.4	1.31950 79107 72894
0.005	1.00347 17485 09503	0.05	1.03526 49238 41377	0.5	1.41421 35623 73095
0.006	1.00416 75432 38973	0.06	1.04246 57608 41121	0.6	1.51571 65665 10398
0.007	1.00486 38204 23785	0.07	1.04971 66836 23067	0.7	1.62450 47927 12471
0.008	1.00556 05803 98468	0.08	1.05701 80405 61380	0.8	1.74110 11265 92248
0.009	1.00625 78234 97782	0.09	1.06437 01824 53360	0.9	1.86606 59830 73615

$10^{\pm n}$ IN OCTAL

10^n	n	10^{-n}	10^n	n	10^{-n}
1	0	1.000 000 000 000 000 000 000 00	112 402 762 000	10	0.000 000 000 006 676 337 66
12	1	0.063 146 314 631 463 146 31	1 351 035 564 000	11	0.000 000 000 000 537 657 77
144	2	0.005 075 341 217 270 243 66	16 432 451 210 000	12	0.000 000 000 000 043 136 32
1 750	3	0.000 406 111 564 570 651 77	221 411 634 520 000	13	0.000 000 000 000 003 411 35
23 420	4	0.000 032 155 613 530 704 15	2 657 142 036 440 000	14	0.000 000 000 000 000 264 11
303 240	5	0.000 002 476 132 610 706 64	34 327 724 461 500 000	15	0.000 000 000 000 000 022 01
3 641 100	6	0.000 000 206 157 364 055 37	434 157 115 760 200 000	16	0.000 000 000 000 000 001 63
46 113 200	7	0.000 000 015 327 745 152 15	5 432 127 413 542 400 000	17	0.000 000 000 000 000 000 14
575 360 400	8	0.000 000 001 257 143 561 06	67 405 553 164 731 000 000	18	0.000 000 000 000 000 000 01
346 545 000	9	0.000 000 000 104 560 276 41			

$n \log_{10} 2$, $n \log_2 10$ IN DECIMAL

n	$n \log_{10} 2$	$n \log_2 10$	n	$n \log_{10} 2$	$n \log_2 10$
1	0.30102 99957	3.32192 80949	6	1.80617 99740	19.93156 85693
2	0.60205 99913	6.64385 61898	7	2.10720 99696	23.25349 66642
3	0.90308 99870	9.96578 42847	8	2.40823 99653	26.57542 47591
4	1.20411 99827	13.28771 23795	9	2.70926 99610	29.89735 28540
5	1.50514 99783	16.60964 04744	10	3.01029 99566	33.21928 09489

ADDITION AND MULTIPLICATION TABLES

Addition

Multiplication

Binary Scale

$$0+0=0 \qquad 0+1=1+0=1 \qquad 1+1=10$$

$$0\times 0=0 \qquad 0\times 1=1\times 0=0 \qquad 1\times 1=1$$

Octal Scale

0	01	02	03	04	05	06	07
1	02	03	04	05	06	07	10
2	03	04	05	06	07	10	11
3	04	05	06	07	10	11	12
4	05	06	07	10	11	12	13
5	06	07	10	11	12	13	14
6	07	10	11	12	13	14	15
7	10	11	12	13	14	15	16

1	02	03	04	05	06	07
2	04	06	10	12	14	16
3	06	11	14	17	22	25
4	10	14	20	24	30	34
5	12	17	24	31	36	43
6	14	22	30	36	44	52
7	16	25	34	43	52	61

MATHEMATICAL CONSTANTS IN OCTAL SCALE

π =	3.11037 552421$_8$	e =	2.55760 521305$_8$	γ =	0.44742 147707$_8$
π^{-1} =	0.24276 301556$_8$	e^{-1} =	0.27426 530661$_8$	$\ln \gamma$ = $-$	0.43127 233602$_8$
$\sqrt{\pi}$ =	1.61337 611067$_8$	$\sqrt{e}$ =	1.51411 230704$_8$	$\log_2 \gamma$ = $-$	0.62573 030645$_8$
$\ln \pi$ =	1.11206 404435$_8$	$\log_{10} e$ =	0.33626 754251$_8$	$\sqrt{2}$ =	1.32404 746320$_8$
$\log_2 \pi$ =	1.51544 163223$_8$	$\log_2 e$ =	1.34252 166245$_8$	$\ln 2$ =	0.54271 027760$_8$
$\sqrt{10}$ =	3.12305 407267$_8$	$\log_2 10$ =	3.24464 741136$_8$	$\ln 10$ =	2.23273 067355$_8$

APPENDIX G

POWERS OF TWO

POWERS OF TWO	2^n	n	n^{-2}
	1	0	1.0
	2	1	0.5
	4	2	0.25
	8	3	0.125
	16	4	0.062 5
	32	5	0.031 25
	64	6	0.015 625
	128	7	0.007 812 5
	256	8	0.003 906 25
	512	9	0 001 953 125
	1 024	10	0.000 976 562 5
	2 048	11	0.000 488 281 25
	4 096	12	0.000 244 140 625
	8 192	13	0.000 122 070 312 5
	16 384	14	0.000 061 035 156 25
	32 768	15	0.000 030 517 578 125
	65 536	16	0.000 015 258 789 062 5
	131 072	17	0.000 007 629 394 531 25
	262 144	18	0.000 003 814 697 265 625
	524 288	19	0.000 001 907 348 632 812 5
	1 048 576	20	0.000 000 953 674 316 406 25
	2 097 152	21	0.000 000 476 837 158 203 125
	4 194 304	22	0.000 000 238 418 579 101 562 5
	8 388 608	23	0.000 000 119 209 289 550 781 25
	16 777 216	24	0.000 000 059 604 644 775 390 625
	33 554 432	25	0.000 000 029 802 322 387 695 312 5
	67 108 864	26	0.000 000 014 901 161 193 847 656 25
	134 217 728	27	0.000 000 007 450 580 596 923 808 125
	268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
	536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
	1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
	2 147 483 848	31	0.000 000 000 465 661 287 307 739 257 812 5
	4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
	8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
	17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
	34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 081 25
	68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
	137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
	274 877 906 944	38	0 000 000 000 003 637 978 807 091 712 951 660 156 25
	549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1	099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2	199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4	398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8	796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17	592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35	184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70	368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140	737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281	474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562	949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125	899 906 842 634	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251	799 813 985 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503	599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 668 164 062 5
9 007	199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 834 582 031 25
18 014	398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 171 513 417 041 015 625
36 028	797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 708 520 507 812 5
72 057	594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 215 395 854 260 253 906 25
144 115	188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 927 130 126 953 125
288 230	376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 963 565 063 476 562 5
576 460	752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 782 531 738 281 25
1 152 921	504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 891 265 869 140 625

APPENDIX H

TABLES OF OCTAL-DECIMAL CONVERSION

TABLE	TITLE
H-1	Octal-Decimal Integer Conversion
H-2	Octal-Decimal Fraction Conversion

Table H-1 Octal-Decimal Integer Conversion Table

0000 to 0777 (Octal) | 0000 to 0511 (Decimal)

Octal Decimal
10000 - 4096
20000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal) | 0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

Table H-1 Octal-Decimal Integer Conversion Table (continued)

2000 to 2777 (Octal)	1024 to 1535 (Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

3000 to 3777 (Octal)	1536 to 2047 (Decimal)

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

Table H-1 Octal-Decimal Integer Conversion Table (continued)

Octal	Decimal
4000 to 4777	2048 to 2559
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

Octal	Decimal
5000 to 5777	2560 to 3071

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

Table H-1 Octal-Decimal Integer Conversion Table (continued)

6000 to 6777 (Octal) | 3072 to 3583 (Decimal)

Octal Decimal
10000 · 4096
20000 · 8192
30000 · 12288
40000 · 16384
50000 · 20480
60000 · 24576
70000 · 28672

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

7000 to 7777 (Octal) | 3584 to 4095 (Decimal)

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

Table H-2 Octal-Decimal Fraction Conversion Table

Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

Table H-2 Octal-Decimal Fraction Conversion Table (continued)

Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

Table H-2 Octal-Decimal Fraction Conversion Table (continued)

Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001265	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

DEC PRODUCT SUMMARY

DIGITAL MODULES

DIGITAL is one of the world's largest suppliers of digital circuit modules. DIGITAL modules have been used in computers, interfaces and special-purpose systems since 1958.

The two major module lines are the M Series TTL integrated circuit modules and the K Series low-speed noise-immune logic. The M Series line has now been expanded to over 60 modules. The K Series product offering has also been expanded, and now comprises nearly 70 modules, all compatible with M Series. Complementing its basic logic, expanded K series offers specialized functional modules such as sensor converters, communication interfaces, 120 V ac interfaces, drivers for solenoids and motors, and logic level converters for tying either high-speed or noise-sensitive logic to other devices.

As the cost of the logic itself decreases, it becomes increasingly important that efficient, reliable and inexpensive hardware be available to keep total system costs down. DIGITAL provides this hardware. Now, from a few to thousands of modules can be connected, wired, mounted, powered and enclosed efficiently with the lowest-cost-per-function in the industry.

DIGITAL's complete line of power and hardware accessories provides everything needed to put your designs into action, from connector blocks to mounting cabinets. Seven power supplies, six connector block variations, ten mounting panels, twelve blank module configurations and five types of connector cards are among the well-over 100 different hardware and accessory items offered by DIGITAL.

PDP-10

PDP-10 is an expandable, 36-bit computer system designed to perform conversational time-sharing, real-time applications, and batch processing simultaneously. It is currently serving in such wide spread applications as physics and bio-medical research, process control, university and industrial computation, time-sharing utilities, chemical research, and hybrid simulation.

The PDP-10 includes an extremely powerful processor with 16 general purpose registers for use as accumulators or index registers, from 16,384 to 262,144 words of 36-bit core memory, and a seven level priority interrupt subsystem. The PDP-10 features an I/O bus which provides 200K word/sec transfer rate and interfaces up to 128 devices or device controllers with the processor. It has 366 easy-to-learn and logically complete instructions.

All PDP-10 systems have in common two basic hardware elements: the central processor and core memory. The same central processor is used in every PDP-10 configuration, but core memory can be composed of any mix of several modules, which vary in size, speed, and cost.

The remainder of the system depends upon the functions it will perform. Software choices include three levels of monitors: single user, multiprogramming, and the multiprogramming/swapping monitor. Most software is re-entrant so that a single compiler can serve many users simultaneously, allowing more users to reside in core at the same time and providing better system response.

Other software includes FORTRAN IV, BASIC, AID, COBOL, a macro assembler, a text editor, a symbolic debugging program, a peripheral interchange program, a desk calculator and FORTRAN library programs. All software systems assure upward compatibility from the standard 16,384 words of memory through the multiprogramming and swapping systems at both the symbolic and relocatable binary level.

Hardware choices include three types of disk systems, magnetic tape units, DECtape systems, line printers, card readers and card punches, communications equipment, CRT displays, plotters, and real-time interface equipment.

PDP-10 features a 1-microsecond cycle time, a 2.6 microsecond add time and a modular, proven software package that expands to make full use of all hardware configurations. Memory can be expanded in 8,192, 16,384, or 32,768 word increments to the maximum directly addressable 262,144 words.

PDP-11

The PDP-11 System is an expandable, 16-bit computer system that is Digital Equipment Corporation's response to industry demands for a radically new concept in computer design. The system contains a variable instruction length processor which directly addresses all of core memory. All communication among system components (including processor and core memory) and peripherals is performed on a single high-speed bus called the Unibus. Because of the bus concept, device-to-device transfers can be accomplished at the rate of 1,300,000 words per second.

With the Unibus concept, devices are modular subsystems physically and electrically attached to the Unibus in parallel. In this configuration, instructions may operate directly upon information in device registers and data transfers from input to output devices can bypass the processor completely. Because of device-memory identity on the Unibus, I/O instructions are not required. A new dimension is added to I/O operation by allowing all instructions to be used with I/O activity.

The PDP-11 processor contains eight hardware registers which are used as arithmetic accumulators, index registers, autoincrement and autodecrement registers. The instruction complement uses the flexibility of these general registers to provide over 400 hard-wired instructions. Because of the flexibility of hardware registers, address modes, instruction set, and direct memory access, PDP-11 programs can be written in directly relocatable codes and every memory location can be used as an accumulator. The processor also includes a full complement of instructions that manipulate byte operands.

The following is a brief list of some of the PDP-11's additional features:

- Interrupts—four levels of hardware interrupt with multiple sublevels. Programmable processor priority level relative to device interrupt requests. Vectored input/output interrupts with unique responses for each device to eliminate polling.
- Stacking—automatic stacking by hardware stack pointer. Any general register can be used to build last-in, first-out stacks. Stack handling capability permits save and restore of program counter in conjunction with subroutine calls and interrupts. This allows true reentrant codes and automatic nesting of subroutines.
- Master/Slave Relationship—communication between any two devices on the bus is interlocked. Any control signal issued by the master must be acknowledged by the slave and vice versa. Therefore, device communication is independent of physical bus length or device response time.
- Asynchronous Bus Operation—processor and Unibus operate asynchronously. Therefore, devices (such as core memory, disk) may bypass processor completely. Processor proceeds independently of operations being performed between devices.
- Direct Memory Access—any number of DMA devices may be attached to the Unibus with priority level among them determined by their placement along the bus. No expensive multiplexers are required.

PDP-12

Digital's PDP-12 is a general-purpose computer system. It is designed as a simple to operate, yet uniquely flexible tool for a wide variety of research and real-time data handling applications.

Performance characteristics of the PDP-12 have been optimized around a complete hardware/software system, rather than an expandable minimum configuration. The PDP-12 systems concept works to the advantage of users at all levels of programming sophistication. By simplifying programming tasks, the PDP-12 frees users from the mere mechanics of program preparation to concentrate on the more creative aspects of their work.

The following is a brief list of some of the PDP-12's outstanding features:

- All-new, unified display-based programming system
- Automatic program loading from magnetic tape
- Built-in program debugging hardware
- 7" x 9" CRT display with graphic and alphanumeric capabilities
- Large existing library of applications programs
- TTL integrated-circuit modules throughout
- LINCtape-addressable, bi-directional program and data storage
- Free-standing cabinet and console table

Specifications:

- 4,096 12-bit words of core memory, expandable to 32,768 words

- 1.6 microsecond cycle time
- 43 basic instructions including 29 memory reference instructions
- 15 auto-index registers
- 6 programmable SPDT relays
- 6 sense switches
- 12 external sense lines
- Peripherals including 16-channel A/D converter, DECtape units, and new 7" x 9" display, all fully buffered
- Software: new unified display-based system; FORTRAN, FOCAL, BASIC, mathematical, maintenance, and utility routines

PDP-15

PDP-15 systems offer comprehensive solutions to real-time data problems. They combine new design concepts with a wide array of traditional features that spring from Digital's years of leadership in the medium-scale scientific computer field. Both elements share the common purpose of simplifying the user's tasks in a demanding real-time environment.

Since certain types of data-handling tasks require specific hardware and software configurations, Digital has developed four standard PDP-15 systems, ranging in power from the modestly priced basic PDP-15/10 to the real-time disk monitor environment of the PDP-15/40. At every level, the capabilities of the hardware are under the control of a monitor designed specifically for them, so that for every step of hardware growth there is a straightforward step of software control to match.

- PDP-15/10: 4,096 18-bit words of 800-nanosecond core memory, Teletype Model ASR-33 console teleprinter.
  COMPACT Software System including assembler, editor, debugging aids, and mathematical and utility routines.

- PDP-15/20 Advanced Monitor System: 8,192 words of core memory, KSR-35 Teletype for extra reliability, two DECtape transports and control unit, high speed paper tape reader/punch, Extended Arithmetic Element for high-speed arithmetic operations and register manipulation. Advanced Monitor System with FORTRAN IV, FOCAL-15, MACRO-15 macro assembler, linking loader, batch processor, system generator, scientific library, and comprehensive debugging and utility routines.

- PDP-15/30 Background/Foreground System: 16,384 words of core memory, KSR-35 Teletype, Extended Arithmetic Element, Automatic Priority Interrupt system, Memory Protect system, high-speed paper tape reader/punch, three DECtape transports and control unit real-time clock, and a second on-line Teletype for backkground use.
  Background/Foreground Monitor System combining all Advanced Monitor functions with concurrent execution of real-time foreground tasks and program development or other low-priority background computation.

- PDP-15/40 Disk-Oriented Background/Foreground System: 24,576 words of core memory, KSR-35 Teletype, Automatic Priority Interrupt system. Relocation/Protect system, high-speed paper tape reader/

punch, two DECtape transports and control unit, RF15 DECdisk control and two RS09 random-access disk files, real-time clock, and a second on-line Teletype for background use.
Disk-oriented version of the Background/Foreground Monitor system, allowing concurrent execution of real-time tasks and background computation, in addition to all standard Advanced Monitor functions.

COMPUTER LAB

The COMPUTER LAB is a new high performance low-cost digital logic trainer. The COMPUTER LAB uses the same monolithic integrated transistor-transistor logic circuitry used in DIGITAL's latest PDP computers.

The digital logic fundamentals presented by the COMPUTER LAB constitute the basic knowledge required to pursue a career in computer technology as computer technician, engineer, programmer or operator. The COMPUTER LAB will also help the math-oriented student to understand "New Math" concepts because computer logic operates with binary numbers according to Boolean algebraic laws.

Wiring is easy because of the standard logic symbology used on the front panel and the color coded Patchcords which are easily inserted and removed. An improper circuit will not damage the COMPUTER LAB. The faulty circuit merely "waits" for correction.

Features:

- Transistor—Transistor logic circuitry as used in DIGITAL's PDP computers
- Teaches modern computer logic
- Easy to use: MIL-STD 806 logic symbology on front panel
- Portable: Dimensions of 12½" x 17" x 3¼", weighing only 11 lbs.
- Comprehensive Workbook provides:
  - —Ten detailed chapters
  - —More than 30 experiments
  - —Over 200 hours of laboratory study
  - —Dozens of tables and diagrams
  - —An extensive appendix of supplementary information
- Teacher's Guide with answers, additional text, extra problems, course plans, at only $5.00
- Low cost: COMPUTER LAB, Workbook and Patchcord set, ready to use $445.00

ALPHABETICAL INDEX

— A —

AA05/AA07 Calibration Tape A-20
AA05/AA07 Digital-to-Analog Converter 7-121
AA50 Digital-to-Analog Converter 7-119
Absolute Address 4-1
Accumulator (AC) 1-4
AD01A 12-bit Analog-to-Digital Converter 7-108
AD08 Diagnostic A-19
AD8-EA Analog-to-Digital Converter 7-65
A/D Calibration Check A-19
A/D Panel, AM8-EC/D 7-73
Address 4-1
Address Modification 4-18
Addressing of DEVICE 9-21
Addressing, Indirect 4-5
Addressing, Memory 4-1
Addressing, Memory, Manual 2-12
Additional DEC Options, Installation of 12-2
AF04A Diagnostic and Demonstration A-19
AF04A Guarded Scanning Integrating Digital Voltmeter 7-114
AFC-8 Low-Level Analog Multiplexer 7-111
ALGOL-8 4-35
Algorithmic Language (ALGOL) 4-35
Alphanumeric Message Typeout A-11
AM8-EC A/D Panel 7-13
AM8-ED A/D Panel 7-73
Analog-to-Digital Converter, AD01A 7-108
Analog-to-Digital Converter, AD8-EA 7-65
Analog Multiplexer, AM8-E 7-68
Analog Multiplexer, Low-Level, AFC-8 7-111
AND, Logical 4-11
Application Programs 4-38
Arithmetic Operations 4-10
ASR 33 Teletype 7-27
Assembler, PAL III 2-28
Assembler Programs A-3
  8K SABR Assembler A-3
  PAL III Assembler A-4
  MACRO-8 Assembler A-4
  PAL-D Disk Assembler (4K) A-4
  PAL-D Disk Assembler (8K) A-4
Assembler Programs 4-32
Assembling a Symbolic Program 2-28
Assigning Locations 4-15
Augmented Instructions 3-5
Autoindexing 4-22
AX08 Diagnostic A-21

— B —

BA8-AA System Expander Box 7-1
BA8-AB System Expander Box 7-1
BASIC 4-36
BASIC-8 A-7
Basic Exerciser, TC01 A-17
Basic Interface Control Circuits, Design of 9-26
Basic Memory Checkerboard A-15
Basic PDP-8/E System 1-2
Basic Programming, Memory and Processor 4-1
BB08-P General Purpose Interface Unit 7-74
BCC Accumulation 7-56
BCD to Binary Conversion Subroutine A-11
BE8-A OMNIBUS Expander 7-11
BIN 2-17
BIN Loader 2-17
Binary Loader 4-37
Binary Loader A-8
Binary Punch A-10
Blank Filler Panels, H945-BA/BB 7-73

Block Check Character (BCC) Accumulator 7-56
Block Diagram, PDP-8/E System 11-14
Board Layout 9-24
Bootstrap Loader, MI8-E 7-17
Bootstrap Loader, TC01 ..... A-8
Branching, Program 4-23
Buffered Digital I/O, DR8-EA 7-71
Bus Data Interface 10-22
Bus Driver Module, M624 .. 10-22
Bus, External 10-1
Bus Lengths, Maximum 10-30
Bus Loads 1-6
Bus Specifications, OMNIBUS 9-4
Bus Structure, OMNIBUS 9-3

— C —

Cable Rules and Suggestions 9-28
Cabling Rules and Suggestions 10-30
Calibration Check, A/D A-19
Calibration Tape, AA05/AA07 A-20
Card Reader, CR8-E 7-60
Card Reader, Optical Mark, CM8-E 7-65
Card Reader Options 7-60
Central Processor Memory Address (CPMA) Register . 1-4
Character Generator, VA8-E .. 7-41
Character Generator, Variable Stroke A-10
Character String Typeout ... A-12
Checking Data Content in Memory 2-12
Clock, Crystal, DK8-EC 7-18
Clock, Line Frequency, DK8-EA 7-18
Clock Panel, DK8-EF 7-73
Clock, Programmable, DK8-EP 7-19
Clock, Real Time, Options 7-17
Clock Timing 1-6
CM8-E Optical Mark Card Reader and Control 7-60
Code Tables C-1
Codes, Device 9-22
Coding a Program 4-15
Combined Microinstructions, Order of Execution 4-25
Combining Microinstructions 4-25
Compatibility Test, Incremental Tape A-18
Compiler Programs 4-34
Compiler Programs A-4
FORTRAN (4K) A-4
FORTRAN (8K) A-5
DIBOL A-5
Computer Internal Options 7-1
Computer Organization 1-2
Configurator, OMNIBUS 9-4
Configuration, Systems 9-4
Console, Programmer's 1-7
Console Teleprinter Control, KL8-E 7-28
Console TTY 7-27
Controls & Indicators 2-1
Control Circuits, Basic Interface, Design of 9-26
Controls, Console Teleprinter, KL8-E 7-28
Control Module, Interface, Design of 9-23
Control, Oscilloscope Display, VC8-E 7-37
Controller, Digital Universal 7-122
Controls, Register 1-5
Control, Teletype 1-7
Conversion, Decimal to Binary A-13
Conversion, Octal-Decimal H-1
Conversion Programs 4-37
Conversion Subroutine, BCD to Binary A-11
Converter, AD01A Analog-to-Digital 7-108
Converter, Analog-to-Digital, AD8-EA 7-65
Converter, Digital-to-Analog, AA05A/AA07 7-121
Converter, Digital-to-Analog, AA50 7-119
Cooling Mounting Panel 10-29
Cosine Routine A-9
Course Descriptions,
DEC Training Program 12
Basic Programming 12-3
PDP-8 Family Programming (Paper-Tape System) ... 12-4
PDP-8 Family Programming (DECtape/Disk System) 12-4
DPD-8/E, 8/I, 8/L Hardware, Familiarization 12-4
CR8-E Card Reader and Control 7-60
CRT Displays 7-37

Crystal Clock, DK8-EC 7-18
Current Address 6-2
Current Page 4-1
Cursor Controller, Joystick, H306 7-45
Customer-Designed Interfaces 10-27
Customer Peripherals 10-19
Cyclic Redundancy 7-56

— D —

Data Acquisition Peripherals 7-108
Data Break Configurations 6-3
Data Break Interface 10-12
Data Break Interface, KD8-E 7-76
Data Break System 6-1
Data Break Timing 9-32
Data Break Transfers 10-12
Data Checking Contents of 2-17
Data Communications Equipment Options 7-52
Data Conversion Programs 4-37
Data Field Register (DF) 7-7
Data Input, Manual Operation 2-17
Data Input, Program Control 2-12
Data Modification Manual 2-12
Data Paths 9-20
Data Register, Data Breaks 6-2
Data Reliability Test, TC58 A-20
Data Storage Manual 2-12
Data Transfer 9-19
Data Transfer Gating 7-7
Data Transfer Timing 9-21
Data Transfers, Programmed I/O 5-4
DDT-8 (Dynamic Debugging Tape) 4-36, A-8
Debugger Programs 4-36
Debugger Programs A-7
DDT-8 A-8
ODT-8 A-8
DEC Training Program 12-2
Course Offerings 12-2
Course Objectives 12-2
Programming 12-3
Hardware Familiarization 12-3
Systems Maintenance 12-3
Course Descriptions 12-3
Decimal to Binary Conversion and Typeout 4-12
DECtape Control, TC08 7-95
DECtape Control, TC08-P 7-97
DECtape Copy Routine A-14
DECtape Format 7-89
DECtape Formatter, TC01/TU55 A-13
DECtape Programming Manual A-13
INDEX A-13
UPDATE A-13
GETSYS A-13
DELETE A-13
DECtape System Software A-13
DECtape Transport, TU56 7-95
DECUS Library 4-38, A-1
Delay Guidelines 9-33
Delays, Program 4-23
DELETE A-13
Design of Interface 9-23
Device Addressing 9-21
Device Codes 9-22
Device Selector and Instruction Decoding 7-7
Device Selector Module, M103 10-20
Defer State 1-9
Depot Repair 12-2
DF32 Disk Data A-15
DF32D Disk File and Control 7-85
DF32 Diskless Logic Test, Minidisk A-21
Digital-to-Analog Converter, AA05A/AA07 7-121
Digital-to-Analog Converter, AA50 7-119
Digital Controller, Universal 7-122
Disk Devices, Random Access 7-77
Disk File Expander, DS32D 7-85
Disk System, RK8 7-78
Diagnostic, AD08 A-19
Diagnostic, AF04A A-19
Diagnostic, AX08 A-21
Diagnostic Programs 4-38, A-14
DIBOL A-5
DIBOL Software System 4-34
Digital Equipment Business-Oriented Language (DIBOL) 4-34
Digital Incremental Plotter, XY8-EA/EB 7-48
Digital I/O, DR8-EA 7-71
Digital Voltmeter, Integrating, AF04A 7-114
Disk Assembler, PAL-D A-4
Disk Data, DF32 A-19

Disk Data, RF08 . . . . A-19
Disk Drive, RK01 7-78
Disk File and Control, DF32D 7-85
Disk Interface Control, RK08-P 7-79
Disk Monitor System . . . 4-31, A-2
Diskless Logic Test, DF32 . A-21
Display Control, Oscilloscope, VC8-E 7-37
Display Control, VS8-E 7-41
Display Diagnostic, KV8/I A-21
Displays, CRT 7-37
Divide, Signed, Subroutine . . A-9
DK8-EA Line Frequency Clock 7-18
DP8-EA/EB Synchronous Modem Interfaces 7-52
DK8-EC Crystal Clock 7-18
DK8-EP Programmable Clock 7-19
Clock Enable Register 7-19
Clock Buffer 7-19
Clock Counter 7-19
Crystal Clock 7-19
External Input Channel 7-19
Programmable Time Base 7-19
DP8-EP Redundancy Check Option . . 7-56
DR8-EA 12-Channel Buffered Digital I/O . . 7-71
Drive Function Time, TC58 A-20
DS32D Disk File Expander 7-85
Dual DECtape Transcript, TU56 7-95
Dynamic Debugging Technique (DDT) 4-36

— E —

EDGRIN A-7
Editor Program 4-32, A-3
Editor Programs
Editor, Symbolic 2-20, A-3
Effective Address 4-1
8K SABR 4-32
Enable Event Interrupt Flip-Flop 7-20
Environmental Requirements 11-6
Equipment Support Service . 12-1
Error Detection 2-26
Etched Circuit Layout and Construction 9-25
Exclusive OR 4-12
Execute State 1-9
Exerciser for Master and Slave Units A-19
Exerciser, TC58 . . . . . A-20
Extended Address Gating . . 7-7
Extended Arithmetic Element . 7-2
Extended Memory Address Test A-15
Extended Memory Checkerboard A-15
Extended Memory Control A-15
Extended Memory Exerciser, TC01 . . . . . A-17
Extended Memory Parity Test A-18
External Bus Connector Pin Assignment 10-18
External Bus, Description . . . 10-1
External Bus Input/Output Equipment Options 7-73
External Bus Interfacing . . . 1-10, 10-1
External Bus Interface Options 7-73
External Bus/OMNIBUS Relationship 9-6
External Bus Signals 10-2

— F —

Family-of-8 33/35 ASR A-16
Family-of-8 High Speed Punch and Reader Tests A-16
Family-of-8 Programs A-1
Fetch State 1-8
Field Installation of Additional DEC Options . . 12-2
Filler Panels, Blank, H945-BA/BB 7-73
Flags 5-3
Flag Testing & Restoration Subroutine 5-6
Floating Point Package 4-38, A-10
Floating Point Package, Four Word A-19
FOCAL-8 . . . . . 4-36, A-7
Four Word Floating Point Package A-9
Formats, Paper Tape 2-9
Formula Calculator (FOCAL) 4-36
FORTRAN A-4
FORTRAN Compiler 4-35
Front Panel, KC8-EB . . 7-1

Front Panel, KC8-EC Turnkey 7-1

— G —

General Description, PDP-8/E 1-1
General Logic Symbols 8-5
General Propagation Delay Guidelines 9-33
General Purpose Interface Unit, BB08-P 7-74
Generating a Program Tape 2-23
GETSYS A-13
Grounding 9-37

— H —

H306 Joystick Cursor Controller 7-45
H945-AA Housing for Mounting Laboratory Peripherals 7-73
H945-BA Blank Filler Panel 7-73
H955-BB Blank Filler Panel 7-73
Hardware Exerciser, TIME-SHARE-8 A-21
Hardware Familiarization Training Program 12-3
HELP Loader A-8
High-Speed Reader Test A-16
Housekeeping Instruction 3-4

— I —

Illegal Combinations, Microprogramming 4-25
Inclusive OR 4-12
Incremental Plotter Control, XY8-E 7-45
Incremental Random Exerciser A-18
Incremental Tape Compatibility Test A-18
Incremental Tape Delay Test A-17
Incremental Tape Instruction Test A-18
INDEX A-13
Indexing Operations 4-13
Indicators & Controls 2-1
Indirect Addressing 4-1
Initializing the PDP-8/E System 2-13
Input/Output Equipment Options 7-27
Input/Output Equipment Options, External Bus 7-73
Input/Output Transfer (IOT) 3-12
Input/Output Transfers, Programmed 10-7
Input Enable Flip-Flop 7-20
Input Flip-Flop 7-20
Input Synchronizers, Programmable Clock 720
Installation Data 11-11
Installation and Planning 11-1
Installation Procedure 11-6
Instruction Buffer Register (IB) 7-7
Instruction Decoding 7-7
Instruction Field Register (IF) 7-7
Instruction Register 1-9
Instruction Summary 3-15
Instruction Test A-14
Instruction Test 1 A-21
Instruction Test 2 A-21
Instruction Test, TC58 A-20
Instructions 3-1
  Augmented 3-5
  Housekeeping 3-4
  Input/Output Transfer 3-12
  Memory Reference 3-3
  Operate 3-5
  Summary 3-15
Instructions, Table of B-1
Interface Control Circuits, Design of 9-26
Interface Control Module, Design of 9-23
Interface, Customer-Designed 10-27
Interface Design 9-23
Interface Logic 10-27
Interface Options, External Bus 7-73
Interface Timing Criteria 9-30
Interface to Bell 201 Modems 7-58
Interface to Bell 301 and 303 Modems 7-59
Interface Unit, General Purpose, BB08-P 7-74
Interfacing, External Bus 1-10, 10-1
Interfacing, OMNIBUS 1-9
Interfacing Techniques 10-16
Internal Options 7-1
Interpretive Programs 4-35
Interpretive Programs A-6

BASIC-8 A-7
EDGRIN A-6
FOCAL-8 A-6
STAND-ALONE BASIC A-6
Interrupt, Program 3-13
Interrupt Subroutine 5-5
Interrupt Timing 9-30
Interrupt Transfers, Programmed 10-12
I/O, Digital, DR8-EA 7-71
I/O Transfers 5-2
IOT Instruction 5-2

— J —

Joystick Cursor Controller, H306 7-45

— K —

KA8-E Positive I/O Bus Interface 7-74
KC8-EB Blank Front Panel 7-1
KC8-EC Turnkey Front Panel 7-1
KD8-E Data Break Interface 7-76
KE8-Extended Arithmetic Element 7-2
Keyboard Operation 2-6
Keyboard/Reader, Teletype 7-29
Keys and Commands, Special 2-26
KL8-E Console Teleprinter Control 7-28
KM8-E Memory Extension and Time-Share Option 7-7
Data Field Register (DF) 7-7
Data Transfer Fating 7-7
Device Selector and Instruction Decoding 7-7
Extended Address Gating 7-7
Instruction Buffer Register (IB) 7-7
Instruction Field Register (IF) 7-7
Save Field Register (SF) 7-7
KM8-E Time Sharing 7-7
KP8-E Power Fail Detect 7-24
KV08/I Display Diagnostic A-21

— L —

Laboratory Peripherals 7-65
Laboratory Peripheral Housing, H945-AA 7-73
Laboratory Peripheral Panel 7-73
LE8 Line Printer 7-48
Library, DECUS 4-38
Light Pen 374A 7-41
Line Frequency Clock, DK8-EA 7-18
Line Printer Character Codes E-1
Line Printer, LE8 7-48
Line Printer Option 7-48
Link (L) 1-4
Loader Programs A-8
Binary Loader A-8
"HELP" Loader A-8
Read-In Mode (RIM) Loader A-8
TC01 Bootstrap Loader A-8
Loader, RIM 2-14
Loader Sequence Perforated Tape D-1
Loaders 2-14, 4-37
Loading and Storing 4-8
Loading Data Under Program Control 2-12
Loading Procedures 2-13
Loading Rules 9-33
Location Assignment 4-15
Logic Circuits 8-1
Logic Operations 4-11
Logical AND 4-11
Logical Shift Routine A-9
Logical Subroutine A-9
Longitudinal Redundancy 7-56
Looping a Program 4-19
Low-Level Analog Multiplexer, AFC-8 7-111

— M —

M101 Bus Data Interface 10-22
M103 Device Selector Module 10-20
M624 Bus Driver Module 10-22
MACRO-8 4-32
MACRO-8 Assembler A-4
Magnetic Tape Options 7-89
MAINDEC A-14
Maintenance and Diagnostic Programs 4-38, A-14
Maintenance and Service Options 12-1
Major Processor States 1-8
Defer State 1-9
Execute State 1-9
Fetch State 1-8
Major Registers 1-4
Accumulator (AC) 1-4
Central Processor Memory Address (CPMA) 1-4

Link (L) 1-4
Memory Buffer (MB) 1-4
Multiplier Quotient (MQ) 1-4
Program Counter (PC) 1-4
Major Register Control Circuits 1-5
Major State Register 1-5
Manual Data Input Operation 2-12
Manual Data Storage and Modification 2-12
Master and Slave Unit Exercises A-19
Master Tape Duplicator A-11
Math Routines Manual A-8
Mathematical Function Routines 4-37
Maximum Operating Frequency 9-35
Mechanical Expansion Options 7-1
Memory Addressing 4-1
Memory Addressing, Manual 2-12
Memory Buffer (MB) Register 1-4
Memory Dump, Octal A-10
Memory Equipment Options 7-7
Memory Extension, KM8-E 7-7
Memory Parity Checkboard A-18
Memory Parity Option 1-8
Memory Parity, MP8-E 7-14
Memory Power On/Off Test A-14
Memory and Processor Basic Programming 4-1
Memory Reference 4-1
Memory Reference Instructions 3-3
Memory Stack 1-7
Memory System 1-7
Message Typeout, Alphanumeric A-11
MI8-E Bootstrap Loader 7-17
Microprogramming 4-25
Minidisk Diskless Logic Test, DF32 ..... A-21
Modification of Address 4-18
Modification of Data, Manual 2-12
Modules, M-Series 10-32
Monitor Programs 4-30
Monitor Programs
Disk Monitor System A-2
PS/8 Programming System A-1
TSS/8 Time-Sharing Monitor A-2
Monitor System, Disk 4-32
Mounting Panel, Cooling of 10-29
Mounting Panel Wiring 10-25
MP8-E Memory Parity Option 7-14
MR8-EA/EB 286-Word Read-Only Memory (ROM) 7-17
M-Series Modules 10-32
Multianalyzer Display and Analysis A-9
Multidisk A-17
Multiplexer, Low-Level Analog, AFC-8 7-111
Multiplier Quotient (MQ) Register 1-4
Multiply, Signed, Subroutine A-9
MW8-E 256-Word Read/Write Memory 7-14

— N —

New Features of PDP-8/E 1-1
9-Track A-20
Notation Scales F-1

— O —

Octal Debugging Technique (ODT) 4-37
Octal-Decimal Conversion H-1
Octal Memory Dump A-10
ODT-8 (Octal Debugging Technique) A-8
OMNIBUS Bus Structure 9-3
OMNIBUS Configurator 9-4
OMNIBUS, Description 9-3
OMNIBUS Expander, BE8-A 7-1
OMNIBUS/External Bus 9-6
OMNIBUS Input/Output Equipment Options 7-27
OMNIBUS Interfacing 1-9, 9-3
OMNIBUS Pin Assignments 9-7
OMNIBUS Signals 9-6
One-Cycle Data Break 6-3
One-Shot Functions 8-4
Operate Instructions 3-5
Group 1 3-5
Group 2 3-8
Group 3 3-10
Operating Procedures for Basic System 2-11
Operations, Programming 4-8
Optical Mark Card Reader, CM8-E 7-65
Options, Internal 7-1
Options, PDP-8/E 7-1
Options, Mechanical Expansion 7-1

Options, Memory Equipment 7-7
OR, Exclusive 4-12
OR, Inclusive 4-12
Order of Execution of Combined Microinstructions 4-25
Organization of Computer 1-2
Oscilloscope Display Control, VC8-E 7-37
Oscilloscope Display, VR14 7-41
Oscilloscope Display, VR03A 7-41
ODT-8 4-37

— P —

Page 4-1
Page Address 4-1
PAL III 2-28, 4-32
PAL III Assembler A-4
PAL-D Disk Assembler A-4
Panel Options 7-1
KC8-EB Blank Front Panel 7-1
KC8-EC Turnkey Front Panel 7-1
Panel, Peripheral Laboratory 7-73
Panel, Real Time Clock, DK8-EF 7-73
Paper Tape Formats 2-9
Paper Tape Punch, PP8-E 7-35
Paper Tape Punch Operation 2-8
Paper Tape Reader, PR8-E 7-33
Paper Tape Reader and Punch Options 7-33
PC8-E Reader/Punch 7-36
PDP-8/E Options 7-1
Perforated-Tape Loader Sequence D-1
Peripherals, Laboratory 7-65
Peripheral Panel, Laboratory 7-73
Peripheral Requirements for Programmed I/O Transfers 5-1
Peripherals, Customer 10-19
Pin Assignment, External Bus Connector 10-18
Pin Assignments, OMNIBUS 9-7
Planning and Installation 11-1
Plotter, Incremental, Control, XY8-E 7-45
Plotter Options, X/Y 7-45
Positive I/O Bus Interface 10-16
Positive I/O Bus Interface, KA8-E 7-74
Power Fail Detect, DP8-E 7-24
Power Fail Test A-15
Power Requirements, External Bus Interfaces 11-6
Power Turn-On Procedure, PDP-8/C 2-12
Powers of Two G-1
PP8-E Paper Tape Punch 7-35
PR8-E Paper Tape Reader 7-33
Printer, Line, LE8 7-48
Printer Operation 2-7
Priority, Data Break 6-2
Program, Branching 4-23
Program Coding 4-15
Program Counter (PC) Register 1-4
Program Delays 4-23
Program Interrupt 3-13
Program Interrupts, Principles of 5-5
Program Looping 4-19
Program Loading Operation 2-13
Programmable Clock, DK8-EP 7-19
Programmed Data Transfer 5-1
Programmed Data Transfer vs. Data Break 5-1
Programmed Interrupt Transfers 10-12
Programmed I/O Transfers 10-7
Programmed I/O Transfers, Principles of 5-2
Programmer's Console 1-7
Programming, Basic, Memory and Processor 4-1
Programming Operations 4-8
Programming Rules 9-38
Programming, System 4-29
Programming Training Program 12-2
Programs, Family-of-8 A-1
Propagation Delay Guidelines 9-33
PS/8 Programming System A-1
Punch, Binary A-11
Punch Options, Paper Tape 7-33
Punch, Paper Tape, PP8-E 7-35
Punch, Read-In Mode A-9
Punch, Service Subroutine 5-7

— R —

Random Access Disk Devices 7-77

Random ADD-ROTATE Test A-14
Random Exerciser, TC01 A-18
Random Exerciser, TC58 A-20
Random ISZ Test A-14
Random JMP Test A-14
Random JMP-JMS Test A-14
Read-In Mode (RIM) Loader 4-37, A-8
Read-In Mode (RIM) Punch A-9
Read-Only Memory (ROM), 256-Word, MR8-EA/EB 7-17
Reader/Punch, PC8-E 7-36
Read/Write Memory, 256-Word, MW8-E 7-16
Reader and Punch Options, Paper Tape 7-33
Reader, Paper Tape, PR8-E 7-33
Reader Service Subroutine 5-7
Real Time Clock Options 7-18
DK8-EA Line Frequency Clock 7-18
DK8-EC Crystal Clock 7-18
DK8-EP Programmable Clock 7-19
Real Time Clock Panel, DK8-EF 7-73
Redundancy Check Option, DP8-EP 7-56
Reference, Memory 4-1
Register Controls 1-5
Instruction Register 1-5
Major Register Control Circuits 1-5
Major State Register 1-5
Requirements, Environmental 11-6
Requirements, Power 11-6
Requirements, Space 11-1
RF08 Disk Data A-19
RIM Loader 2-14, 4-37, A-8
RIM Punch A-9
RK01 Disk Drive and Control 7-78
RK08-P Disk Interface Control 7-79
RK8 Disk System 7-77
ROM (Read-Only Memory) 7-17

— S —

SABR Assembler A-7
Save Field Register (SF) 7-7
Scales of Notation F-1
Schmitt Trigger 8-4
Schmitt Triggers, Specification 7-20
Search Feature Using Editor 2-25
Sense/Inhibit Module 1-8
Service Contract 12-1
Service Contract Options 12-1
Service Options 12-1
7-Track A-20
Shift Left Subroutine A-9
Shift Right Subroutine A-9
Signals, External Bus 10-2
Signals, OMNIBUS 9-6
Signed Decimel Print A-12
Signed Divide Subroutine A-9
Signed Multiply Subroutine A-9
Signal Terminating 10-29
Sine Routine A-9
Skip Microinstructions 4-27
Software, DECtape System A-13
Software Kit 4-29
Space Requirements 11-1
Square Root Subroutines A-9
STAND-ALONE BASIC A-7
State Indicator 8-1
Storage of Data, Manual 2-12
Storage Tube Display, VT01 7-45
Storing and Loading 4-8
Subroutine for Flag Testing & Restoration 5-6
Subroutine for Punch Service 5-7
Subroutine for Reader Service 5-7
Subroutine, Interrupt 5-6
Subroutines, Writing 4-16
Support Services 12-1
Symbolic Assembler, PAL III 2-28
Symbolic Assembler for Binary Relocatable Programs (SABR) 4-34
Symbolic Editor 2-21, A-3
Symbolic Paper Tape Editor 4-32
Symbolic Tape Format Generator A-12
Sync Flip-Flop 7-20
Synchronous Modem Interface, DP8-EA/EB 7-52
System Block Diagram, PDP-8/E 11-14
System Configuration 11-12

System Expander Boxes 7-1
BA8-AA System Expander Box 7-1
BA8-AB System Expander Box 7-1
System Introduction, PDP-8/E 1-1
Systems Maintenance Training Program 12-3
System Programming 4-29
System Programs 4-30
System Programs A-2
Assembler Programs A-8
Compiler Programs A-8
Debugger Programs A-7
Editor Programs A-3
Interpretive Programs A-6
Loader Programs A-8
Monitor Programs A-2
Utility Programs A-8
Systems Configuration 9-4

— T —

Tables of Codes C-1
Tape Duplicator, Master A-11
Tape, Generation of 2-23
Tape Options, Magnetic 7-89
TC01 Basic Exerciser A-17
TC01 Bootstrap Loader A-8
TC01 Extended Memory Exerciser A-17
TC01 Random Exerciser A-18
TC01/TU55 DECtape Formatter A-13
TC08 DECtape Control 7-95
TC08-P DECtape Control 7-97
TC58 Drive Function Timer A-20
TC58 Instruction Test A-20
TC58 Random Exerciser A-20
Teleprinter Control, Console, KL8-E 7-28
Teleprinter/Punch 7-31
Teletype, Console 7-27
Teletype Control 1-7
Teletype Keyboard/Reader 7-29
Teletype Test A-16
Terminating Signals 10-29
Testing Techniques 9-37
Three-Cycle Data Break 6-7
Times, TC58 Drive Function A-20
Timing Criteria, External Bus Interfaces 10-29
TIME-SHARE-8 Hardware Exerciser A-21
Time Share Option, KM8-E 7-7
Time-Sharing Monitor 4-32
Time-Sharing Monitor, TSS/8 A-2
Time-Sharing KM8-E 7-7
Time-Sharing 8 Option Test A-21
Timing Criteria, Interface 9-30
Timing, Clock 1-6
Timing Data Break Facilities 9-32
Timing, Data Transfer 9-21
Timing Generator 1-6
Timing of Interrupts 9-30
Teletype Output Subroutine A-11
Training Program 12-6
TR22, 25, 28 Incremental Tape Instruction Test A-18
Training Program Objectives 12-2
Transfer of Data 9-21
TSS/8 Time-Sharing Monitor A-2
TTY (Teletype) Console 7-27
TU56 DECtape Transport 7-95
Turn-On Procedure, PDP-8/E 2-12
Two, Powers of G-1
Two's Complement Arithmetic 4-10

— U —

UDC (Universal Digital Controller) 7-122
Universal Digital Controller (UDC) 7-122
Unsigned Decimel Print A-12
UPDATE A-13
Utility Programs 4-37
Utility Programs A-8
Cosine Routine, Double Precision A-9
Four Word Floating-Point Package A-9
Logical Shift Routine A-9
Logical Subroutine A-9
Math Routines Manual A-8
Shift Left Subroutine, Single and Double Precision A-9
Shift Right Subroutine, Single and Double Precision A-9
Signed Divide Subroutine, Double Precision A-9

Signed Divide Subroutine, Single Precision A-9
Signed Multiply Subroutine, Double Precision A-9
Signed Multiply Subroutine, Single Precision A-9
Sine Routine, Double Precision A-9
Square Root Subroutine, Single Precision A-9

— V —

VA8-E Character Generator 7-41
Variable Stroke Character Generator (VSCG) A-10
VC8-E Oscilloscope Display Control 7-37
Vertical Redundancy 7-56
Voltmeter, Integrating Digital, AF04A 7-114
VR03-A Model 602 Tektronics Oscilloscope 7-73
VR03A Oscilloscope Display 7-41
VR14 Oscilloscope Display 7-41
VS8-E Display Control 7-41
VT01 Storage Tube Display 7-45

— W —

Wiring of Mounting Panel 10-25
Writing a Program 2-23
Writing Subroutines 4-16
Word Count 6-2

— X —

XY8-EA/EB Digital Incremental Plotter 7-48
XY8-E Incremental Plotter Control 7-45
XY Current Source 1-7
XY Drive and Current Source 1-7
X/Y Plotter Options 7-45

NOTES

NOTES

NOTES

NOTES

NOTES

DIGITAL EQUIPMENT CORPORATION digital WORLD-WIDE SALES AND SERVICE

MAIN OFFICE AND PLANT

146 Main Street, Maynard, Massachusetts, U S A 01754 • Telephone: From Metropolitan Boston: 646-8600 • Elsewhere: (617)-897-5111

TWX: 710-347-0212 Cable: DIGITAL MAYN Telex: 94-8457

UNITED STATES

NORTHEAST

REGIONAL OFFICE
15 Lunda Street, Waltham, Massachusetts 02154
Telephone: (617)-891-1030 TWX: 710-324-0919

WALTHAM
15 Lunda Street, Waltham, Massachusetts 02154
Telephone: (617)-891-6310/6315 TWX: 710-324-0919

CAMBRIDGE/BOSTON
899 Main Street, Cambridge, Massachusetts 02139
Telephone: (617)-491-8130 TWX: 710-320-1167

ROCHESTER
130 Allens Creek Road, Rochester, New York 14618
Telephone: (716)-461-1700 TWX: 710-599-3211

CONNECTICUT
1 Prestige Drive, Meriden, Connecticut 06450
Telephone: (203)-237-8441 TWX: 710-461-0054

MID-ATLANTIC — SOUTHEAST

REGIONAL OFFICE:
U S Route 1, Princeton, New Jersey 08540
Telephone: (609)-452-9150 TWX: 510-685-2338

NEW YORK
95 Cedar Lane, Englewood, New Jersey 07631
Telephone: (201)-871-4984, (212)-594-6955, (212)-738-0447
TWX: 710-991-9721

NEW JERSEY
1258 Route 46, Parsippany, New Jersey 07054
Telephone: (201)-335-3300 TWX: 710-987-8319

PRINCETON
Route One and Emmons Drive,
Princeton, New Jersey 08540
Telephone: (609)-452-2940 TWX: 510-685-2337

LONG ISLAND
1919 Middle Country Road
Centereach, L I , New York 11720
Telephone: (516)-585-5410 TWX: 510-228-6505

PHILADELPHIA
Station Square Three, Paoli, Pennsylvania 19301
Telephone: (215)-647-4900 Telex: 510-668-8395

WASHINGTON
Executive Building
7100 Baltimore Ave , College Park, Maryland 20740
Telephone: (301)-779-1100 TWX: 710-826-9662

DURHAM/CHAPEL HILL
2704 Chapel Hill Boulevard
Durham, North Carolina 27707
Telephone: (919)-489-3347 TWX: 510-927-0912

MID-ATLANTIC — SOUTHEAST (cont.)

ORLANDO
Suite 332, 6990 Lake Ellenor Drive, Orlando, Fla. 32809
Telephone: (305)-851-4450 TWX: 810-850-0180

ATLANTA
2815 Clearview Place, Suite 100,
Atlanta, Georgia 30340
Telephone: (404)-457-7787 TWX: 810-757-4223

KNOXVILLE
5731 Lyons View Pike, S W., Knoxville, Tenn. 37919
Telephone: (615)-588-8571 TWX: 810-583-0123

CENTRAL

REGIONAL OFFICE:
1850 Frontage Road, Northbrook, Illinois 60062
Telephone: (312)-498-2560 TWX: 910-686-0655

PITTSBURGH
400 Penn Center Boulevard
Pittsburgh, Pennsylvania 15235
Telephone: (412)-243-8500 TWX: 710-797-3657

CHICAGO
1850 Frontage Road, Northbrook, Illinois 60062
Telephone: (312)-498-2500 TWX: 910-686-0655

ANN ARBOR
230 Huron View Boulevard, Ann Arbor, Michigan 48103
Telephone: (313)-761-1150 TWX: 810-223-6053

INDIANAPOLIS
21 Beechway Drive — Suite G
Indianapolis, Indiana 46224
Telephone: (317)-243-8341 TWX: 810-341-3436

MINNEAPOLIS
Suite 111, 8030 Cedar Avenue South,
Minneapolis, Minnesota 55420
Telephone: (612)-884-4092 TWX: 910-576-2818

CLEVELAND
Park Hill Bldg , 35104 Euclid Ave
Willoughby, Ohio 44094
Telephone: (216)-946-8484 TWX: 810-427-2608

ST. LOUIS
Suite 110, 115 Progress Pky , Maryland Heights,
Missouri 63043
Telephone: (314)-827-7520 TWX: 910-764-0831

DAYTON
3101 Kettering Blvd , Dayton, Ohio 45439
Telephone: (513)-299-7377 TWX: 810-459-1676

CENTRAL (cont.)

DALLAS
8855 North Stemmons Freeway, Suite 204
Dallas, Texas 75247
Telephone: (214)-638-4860 TWX: 910-861-4000

HOUSTON
3417 Milam Street, Suite A, Houston, Texas 77002
Telephone: (713)-524-2961 TWX: 910-881-1651

WEST

REGIONAL OFFICE
580 San Antonio Road, Palo Alto, California 94306
Telephone: (415)-326-5640 TWX: 910-373-1288

ANAHEIM
801 E. Ball Road, Anaheim, California 92805
Telephone: (714)-776-6932 or (213)-625-7689
TWX 910-591-1169

WEST LOS ANGELES
1510 Cotner Avenue, Los Angeles, California 90025
Telephone: (213)-479-3791 TWX: 910-342-6999

SAN FRANCISCO
560 San Antonio Road, Palo Alto, California 94306
Telephone: (415)-326-5640 TWX: 910-373-1286

OAKLAND
7850 Edgewater Drive
Oakland, California 94621
Telephone: (415)-635-5453, (415)-635-7830
TWX 910-366-7238

ALBUQUERQUE
6303 Indian School Road, N E
Albuquerque, N M 87110
Telephone: (505)-296-5411 TWX: 910-989-0614

DENVER
2305 South Colorado Blvd., Suite #5
Denver, Colorado 80222
Telephone: (303)-757-3332 TWX: 910-931-2650

SEATTLE
1521 130th N E , Bellevue, Washington 98004
Telephone: (206)-454-4058 TWX: 910-443-2308

SALT LAKE CITY
431 South 3rd East, Salt Lake City, Utah 84111
Telephone: (801)-328-9838 TWX: 910-925-5834

PHOENIX
307 E. Southern Ave., Tempe, Arizona 85281
Telephone: (602)-967-1618

INTERNATIONAL

CANADA

Digital Equipment of Canada, Ltd

CANADIAN HEADQUARTERS
150 Rosamond Street, Carleton Place, Ontario
Telephone: (613)-257-2615 TWX: 610-561-1651

OTTAWA
120 Holland Street, Ottawa 3, Ontario
Telephone: (613)-725-2193 TWX: 610-562-8907

TORONTO
230 Lakeshore Road East, Port Credit, Ontario
Telephone: (416)-278-6111 TWX: 610-492-4308

MONTREAL
9675 Cote de Liesse Road
Dorval, Quebec, Canada 760
Telephone: 514-636-9393 TWX 610-422-4124

EDMONTON
5331-103 Street
Edmonton, Alberta, Canada
Telephone: (403)-434-9333 TWX: 610-831-2248

VANCOUVER
Digital Equipment of Canada, Ltd.
2210 West 12th Avenue
Vancouver 9, British Columbia, Canada
Telephone: (604)-736-5618

SOUTH AMERICA

COASIN S A
Virrey del Pino 4071 Buenos Aires, Argentina
Telephone: 52-3185 Telex: 012-2284

EUROPEAN HEADQUARTERS

Digital Equipment Corporation International-Europe
81 Route De L'Aire
1227 Carouge / Geneva, Switzerland
Telephone: 42 79 50 Telex: 22 683

GERMANY

Digital Equipment GmbH

COLOGNE
5 Koeln, Bismarckstrasse 7, West Germany
Telephone: 52 21 81 Telex: 841-888-2269
Telegram: Flip Chip Koeln

MUNICH
8 Muenchen 13, Wallensteinplatz 2
Telephone: 0811-356011-15 Telex: 841-52-4226

GERMANY (cont.)

HANOVER
Digital Equipment Corporation GmbH
3 Hanover, Podbielskistrasse 102
Telephone: 0511-697-095 Telex: 922-952

ENGLAND

Digital Equipment Co., Ltd.

READING
Arkwright Road, Reading, Berkshire, England
Telephone: Reading 85131 Telex: 84327

MANCHESTER
6 Upper Precinct, Worsley
Manchester, England
Telephone: 061-790-8411 Telex: 668668

LONDON
Bilton House, Uxbridge Road, Ealing, London W.5.
Telephone: 01-579-2781 Telex: 22371

FRANCE

Equipment Digital S.A.R.L.

PARIS
327 Rue de Charenton, 75 Paris 12 EME, France
Telephone: 344-76-07 Telex: 21339

BENELUX

Digital Equipment N.V.
(serving Belgium, Luxembourg, and The Netherlands)

THE HAGUE
Koninginnegracht 65, The Hague, Netherlands
Telephone: 635960 Telex: 32533

SWEDEN

Digital Equipment Aktiebolag

STOCKHOLM
Vratenvagen 2, S-171 54 Solna, Sweden
Telephone: 08 98 13 90 Telex: 170 50
Cable: Digital Stockholm

SWITZERLAND

Digital Equipment Corporation S A

GENEVA
81 Route De L'Aire
1227 Carouge / Geneva, Switzerland
Telephone: 42 79 50 Telex: 22 683

SWITZERLAND (cont.)

ZURICH
Freigutstrasse 26, 8002 Zurich
Telephone: (51) 36 78 23

ITALY

Digital Equipment S.p.A.

MILAN
Corso Garibaldi, 49, 20121 Milano, Italy
Telephone: 872 748, 872 694, 872 394 Telex: 33615

AUSTRALIA

Digital Equipment Australia Pty. Ltd.

SYDNEY
75 Alexander St., Crows Nest, N.S.W. 2065, Australia
Telephone: 439-2566 Telex: AA20740
Cable: Digital, Sydney

MELBOURNE
60 Park Street, South Melbourne, Victoria, 3205
Telephone: 69-6142 Telex: AA30700

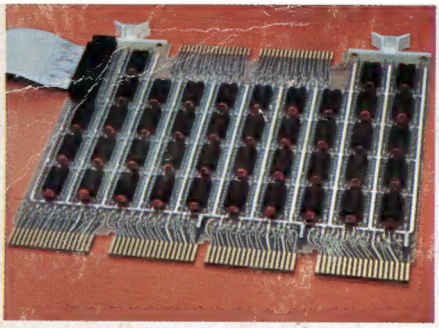
WESTERN AUSTRALIA
643 Murray Street
West Perth, Western Australia 6005
Telephone: 21-4993 Telex: AA92140

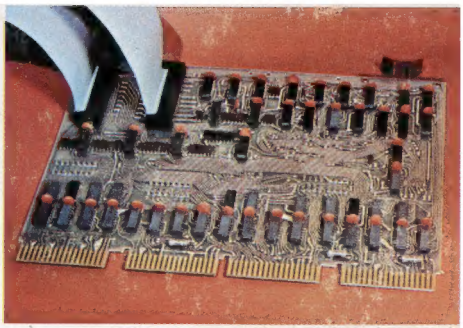
BRISBANE
139 Merivale Street, South Brisbane
Queensland, Australia 4101
Telephone: 44047 Telex: AA40616

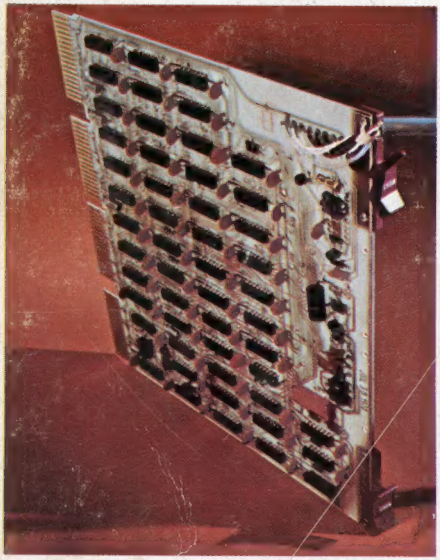
JAPAN

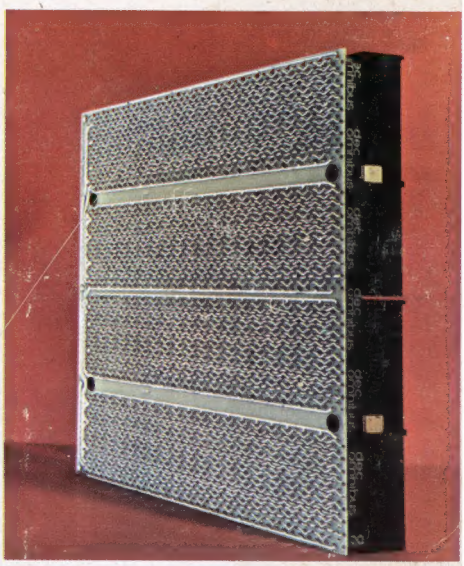
TOKYO
Rikei Trading Co., Ltd. (sales only)
Kozato-Kaikan Bldg
No. 18-14, Nishishimbashi 1-chome
Minato-Ku, Tokyo, Japan
Telephone: 5815246 Telex: 781-4208

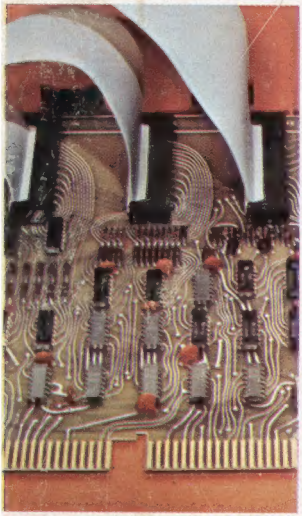
Digital Equipment Corporation International
Kowa Building No. 17, Second Floor
2-7 Nishi-Azabu 1-Chome
Minato-Ku, Tokyo, Japan
Telephone: 404-5894/6 Telex: TK-6428

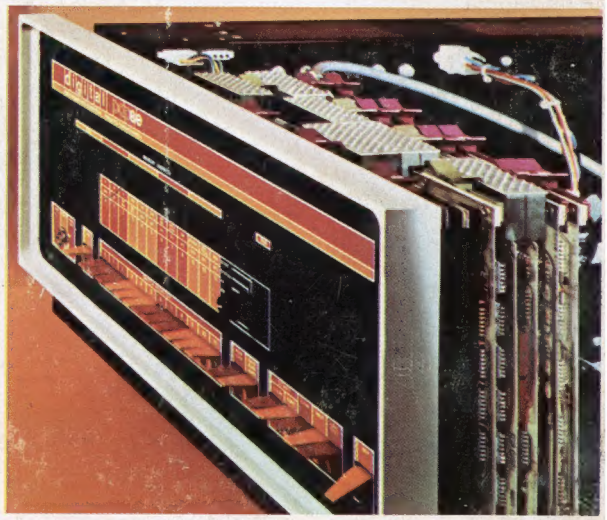












082X • 01170 • AMB

A1127

Printed in U.S.A.